Action recognition using shared motion parts

Alonso Patron-Perez, Eric Sommerlade, Ian Reid

Action recognition using shared motion parts.

Alonso Patron-Perez, Eric Sommerlade and Ian Reid
Department of Engineering Science, University of Oxford
OX1 3PJ, Oxford, UK
{alonso, eric, ian}@robots.ox.ac.uk

Abstract

In this paper we analyse the advantages of a joint boosting method, previously known in the object recognition community, to detect and classify action keyframes. The method focuses on sharing object parts among action classes. Instead of sharing parts that only encode shape similarities, we propose to include motion information as an extra clue for sharing. We show that the inclusion of motion information significantly improves the recognition accuracy. The method is tested using a standard action database containing 10 action classes obtaining perfect classification. It also yields promising results on complicated videos including complex background.

1. Introduction

The detection and classification of human actions is an active field of research. It has a wide range of applications from intelligent surveillance and game applications to video retrieval and content compression. As a result of this, a great variety of methods has been developed following different approaches. These methods try to overcome some of the main difficulties inherent in this task, namely, occlusions, scale changes, moving background, view-invariance, simultaneous actions and camera motion, examples of which can be seen in Fig. 1. Many of these problems have already been addressed in the object recognition community, and we take advantage of this knowledge for the action recognition problem.

One approach is to represent each action by a subset of still frames that best describes it. These frames are called keyframes and represent specific poses of the human body when performing an action. This is a simple but powerful model especially when combined with other cues as we show in this paper. Many of the methods based on keyframes try to match complete human silhouettes which requires accurate segmentation of subjects from the background.

Figure 1. Complex scenes illustrating some of the difficulties of the action detection and classification task.

Another approach in action classification methods is to use parts. In this context by a part we mean an image or video fragment. Part-based models have several inherent advantages over other types of methods: they are fairly robust to occlusions, simple to extract and flexible in their representation. They also give a well-tested framework for detection and classification and for these reasons we choose to work with them. However, as noted by Torralba et al. [19], a disadvantage of this model is that the number of parts needed to represent different classes grows linearly with each new class that is added.

The insight presented in [19] was that many objects share similar parts and that when a new class is added we can reuse previously learned parts to classify the new one, therefore keeping the number of new parts that have to be added to a minimum. We apply this idea to human actions by noticing that different actions not only have similarly shaped parts but also many of the parts move in a common direction. We recognize that although the number of action classes is not a current problem due to the small size of the

available human action databases (which will probably increase in the near future), this method has other advantages like reducing the time needed for detection and recognition, a by-product of having a smaller set of parts to represent an action set.

The contribution of this paper is to show that we can combine keyframe and part-based models by sharing parts among classes. This new combination of existing disparate methods enables us to describe different action classes using a compact representation and achieve high detection and classification accuracy.

The remainder of the paper is organised as follows: section 2 reviews related work on keyframe and part-based models. Section 3 is divided in two parts: the first one presents the specific feature extraction method used in this paper while the second one describes the basics of the joint boosting method designed for the explicit sharing of features among classes. In section 4 we show the results of the classification for a small database and some qualitative experiments in more complex scenes which demonstrate how this method could be extended to deal with such difficult cases.

2. Related work

The use of spatio-temporal parts (video fragments) has become popular in recent years, and several methods have been presented where only the response function used to detect interest locations in the video is changed. Laptev and Lindeberg [10] extracted video fragments around interest points located in regions of motion discontinuity. Dollar et al. [5] used locations where periodic motion was present and Oikonomopoulus et al. [16] used regions of local maximum entropy. Niebles et al. [14] used the same response function as Dollar but changed the classification method using a pLSA (probabilistic Latent Semantic Analysis) approach. More recently Niebles et al. [15] presented a method that combines static features and spatio-temporal features using a hierarchical constellation model. Laptev has gone one step further trying to recognise actions in movies [12, 11] using keyframes to obtain possible location hypotheses and validating these with temporal information. He uses AdaBoost for selecting distinctive video fragments that can be used for classification.

This begs the question: how many frames are needed to describe an action?. This question has been treated recently in [17, 21], where the results show that even with few frames a high classification accuracy can be achieved. Their method does not provide detection; just classification of actions given an already segmented sequence of frames centred on the subject. Action recognition with keyframes has also been tested by Carlsson and Sullivan [2, 18] relying completely on a shape matching algorithm and by

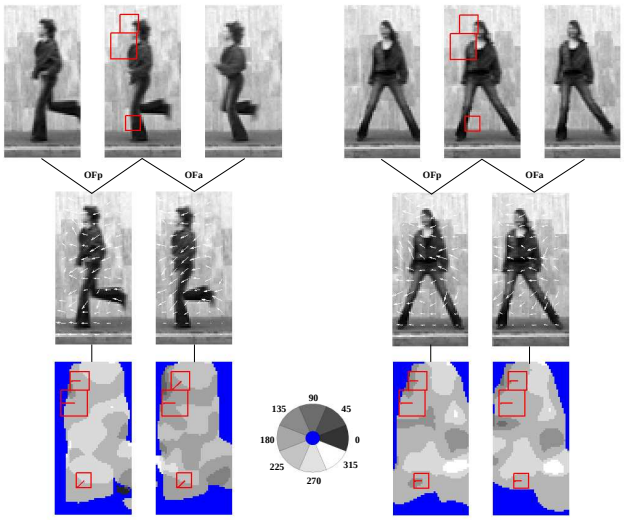

Figure 2. First row shows two different action sequences (running and galloping sideways) with their respective keyframe in the middle. The middle row shows the previous and posterior optic flow calculated using the Lucas-Kanade method. The bottom row shows the discretized optic flow in 8 directions (displayed in gray scale). Also shown are three examples of random patches selected and their respective dominant orientation in each of the actions. As can be seen the patches at the top part of the body could be shared by these two classes because of their simmilarity in shape and motion while the patch located near the legs could be used to discriminate between them.

Efros [6] using a pure motion descriptor. It has been shown [15, 17, 9, 4, 3] that a combination of shape and motion increases the detection and classification performance. Our approach follows the spirit of these papers; we will show that many actions can be described by a single keyframe. The problem with keyframes is that when the number of actions increases, it is harder to find a single frame that can distinguish a particular action from the rest. In this case, motion can be used to disambiguate between them.

3. Shared part model

The main observation made by [19] was that there are parts that can be shared among different object classes, thus greatly reducing the final number of parts needed to learn a classifier. We show that this can be directly applied to human action recognition by adding motion information to

each part; for example, waving with one or two hands, and jumping jack all share similar arm motion.

3.1. Feature vector estimation

Here we describe how we generate a pool of feature vectors from which the boosting method will select an optimal subset during training. First, from a set of action keyframes we select a fixed size region of interest centred around the person performing the action. If the people in the training examples present a significant difference in scale, they have to be normalized to a fixed size. We also extract two optic flow fields between the keyframe and its previous and subsequent frames using the Lucas-Kanade method [13] (Fig. 2). We use the orientation of these flow fields by discretising it to one of n directions if its magnitude is above a given threshold. Then we extract M image fragments randomly from these regions (varying in size from 7x7 to 15x15). For each fragment the following information is also saved: a spatial binary mask (that represents its position with respect to the object centre) and the dominant orientations of the two related optic flow fields that lie directly in the image region that the fragment covers. Dominant orientations are estimated by histogramming the optic flow orientations of every pixel inside the fragment and choosing the bin with the highest frequency.

The feature vector x from a training image is calculated as follows: the jth element of the feature vector is obtained, first, by applying normalized cross correlation (NCC) between the training image and the jth extracted fragment and exponentiating the response to a power p. By using higher values of p we are effectively performing template matching (we will use $p = 10$ in all the experiments as suggested in [19]). Unlike [19] we weight this template matching value by a factor w_{of}, which measures the optic flow similarity between the fragment and all regions of the training image. The weight at a specific image location $\ell = (x, y)$ is a normalized value in $[0, 1]$ (1 meaning no difference and 0 a maximum distance of $180°$) calculated as the difference between the angles of the previous and subsequent dominant optic flow orientations in a region centred at ℓ (and of the same size as the jth fragment) and the respective dominant flow orientations of the fragment. Finally, a convolution with the fragment's spatial mask is performed which can be seen as the fragment voting for a probable object centre. The size of the binary spatial masks allows small translations of the original fragment location. The response of this function at the object's centre is taken to be the jth element of the feature vector; this is expressed in equation 1. We obtain a feature vector for each training image by applying this process with each extracted fragment. The process of estimating the jth element of the feature vector for a training image is illustrated in Fig. 3a (the complete feature vector displaying the response value of all fragments is shown in Fig. 3b).

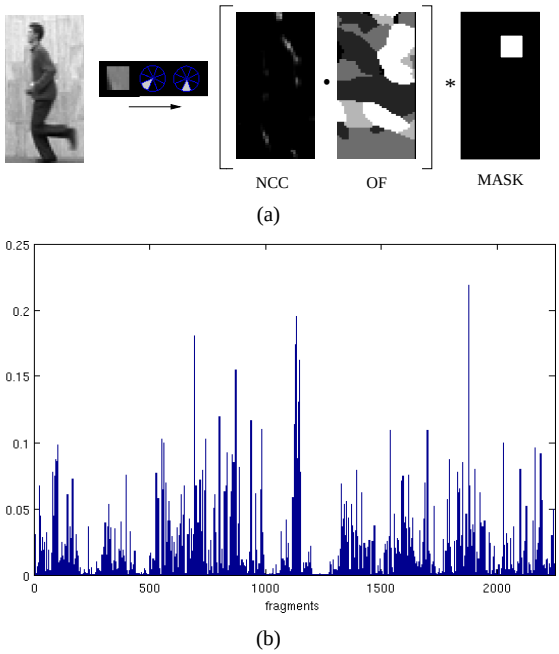

Figure 3. Generation of a feature vector. (a) An example of the process for one training image and one fragment. NCC and Optic Flow (OF) matching maps are computed. The OF is used to weight the NCC matching. Finally a convolution with the fragment's spatial mask is performed. (b) The complete feature vector for the training image in (a). Each element records the response value of one fragment.

In the testing phase, we follow the same procedure, constructing feature vectors at each image location and classifying these as containing one of the action classes or not. For a location ℓ of an image I at scale σ the feature vector is given by:

$$x^j(\ell, \sigma) = m_j(\ell) * (|I_\sigma \otimes g_j|^p \bullet w_{of}) \qquad (1)$$

where $[\otimes * \bullet]$ represent the NCC, convolution and elementwise multiplication operators respectively, g_j is the jth fragment and $m_j(\ell)$ its spatial binary mask centred at location ℓ. As can be seen from the definition of equation 1, the method becomes multiscale by constructing a Gaussian pyramid of the input image and searching in a range of scales. To avoid optic flow estimation errors in smaller scales we only calculate optic flow at the original scale. When estimating the optic flow weight value for a specific region in a smaller image, we look for the weight

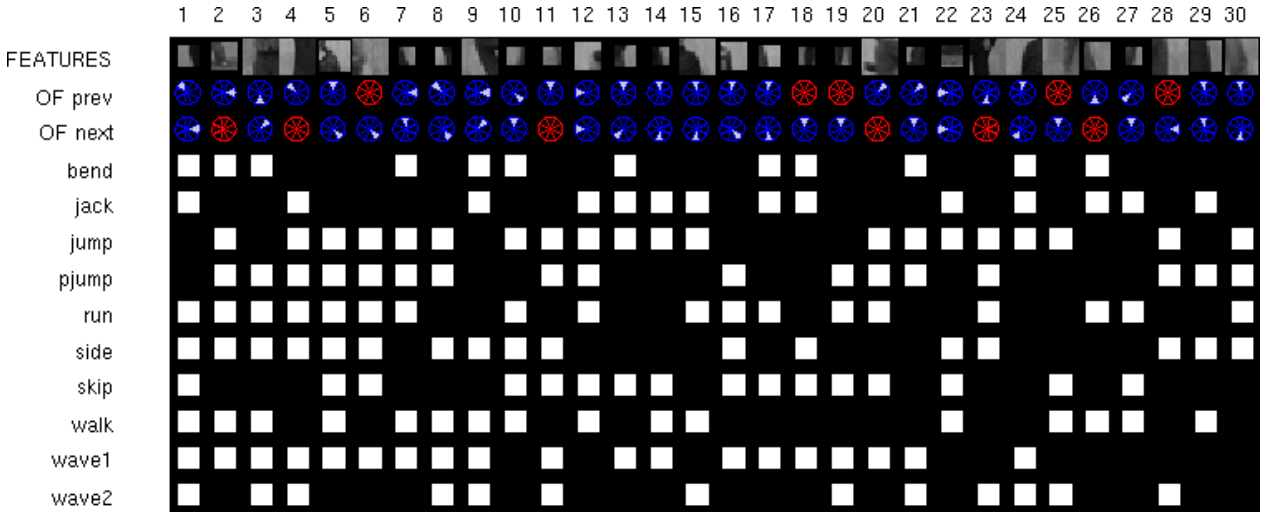

Figure 4. First 30 fragments selected by the boosting algorithm sorted by the number of classes (actions) that share them (marked with a white square). Optic flow orientations associated with the frame before and after the keyframe are shown in the second and third rows.

of the corresponding region in the original scale.

3.2 Joint boosting method

The idea of the joint classifier presented in [19] is based on a modification of the popular AdaBoost (Adaptive Boosting) known as GentleBoost [7]. As in any boosting method, we try to construct a strong classifier $F(x, c)$ by adding up several weak classifiers $f_m(x, c)$ (equation 2), where x is the input data (in our case the feature vector described in the previous section) and c is the class.

$$F(x, c) = \sum_{m}^{M} f_m(x, c) \tag{2}$$

Before getting into more details of the Joint boosting method, we review the learning process of a standard AdaBoost algorithm. At each boosting cycle, a weak classifier is chosen that best classifies the weighted training examples. This weighting is obtained by giving a higher weight to the examples that were misclassified in the previous round, therefore biasing the selection of the next weak classifier to perform better on these erroneous classifications. Given a test example, each weak classifier makes a binary decision whether it belongs to a specific class or not. Normally we obtain a final classification by taking the sign of the sum of the weak classifiers' response.

As we have seen boosting algorithms are binary classifiers, and, when dealing with multiple classes, strong classifiers have to be trained independently for each class. The result of this process is that the selected weak classifiers are very class-specific.

In contrast, the joint classifier trains all strong classifiers simultaneously and is designed to favour more general weak classifiers that can be shared among different classes. This translates to the inclusion of a new criterion for the selection of a weak classifier: we have to check not only the reduction of misclassification error for a single class, but the overall error reduction if we use that same classifier to separate a subset of classes from the rest (including the background). The function that we want to minimise is given in equation 3, where z_i^c are the data labels $\{-1, 1\}$ of the training set, which is an extended version of the regular exponential error function used in many boosting methods (the only difference being the summation with respect to the classes).

$$J = \sum_{c=1}^{C} E[e^{-z_i^c F(x,c)}] \tag{3}$$

The above cost function is minimised similarly to the regular GentleBoost algorithm using adaptive Newton steps [7]. In other words, at each round of the boosting algorithm we choose a feature and the best subset of classes S that minimises the weighted square error:

$$J_{wse} = \sum_{c=1}^{C} \sum_{i=1}^{N} w_i^c (z_i^c - f_m(x_i, c))^2. \tag{4}$$

where $w_i^c = e^{-z_i^c F(x_i,c)}$ are the weights estimated for example i using the strong classifier for class c. For weak classifiers, we use stumps (decision trees with one node),

that consist of a simple thresholding of a random input variable, which for a selected subset of classes S take the form:

$$f_m(x,c) = \begin{cases} a_S & \text{if} \quad x^j > \theta \quad \text{and} \quad c \in S \\ b_S & \text{if} \quad x^j \leq \theta \quad \text{and} \quad c \in S \\ k_S^c & \text{if} \quad c \notin S \end{cases} \tag{5}$$

where x^j represents the jth entry of the input feature vector (see the previous section for an explanation of how this vector is calculated). Another way of understanding the above equation is to consider a weak classifier as a function $f_m(x) = a\delta(x^j > \theta) + b\delta(x^j \leq \theta)$, where a and b are regression parameters. Each weak classifier needs to save the subset of classes S, the parameters (a_S,b_S,θ,j) and a set of class-specific constants k_S^c similar to [19] to avoid selecting a class due to an imbalance among the number of positive and negative examples in the training set. This is done by testing all possible combinations of classes, features and thresholds and choosing the ones that minimise equation 4. Given a subset S a feature j and a threshold θ the values of a_S,b_S and k_S^c that minimise equation 4 are given by:

$$a_S(j,\theta) = \frac{\sum_{c\in S}\sum_i w_i^c z_i^c \delta(x_i^j > \theta)}{\sum_{c\in S}\sum_i w_i^c \delta(x_i^j > \theta)}, \tag{6}$$

$$b_S(j,\theta) = \frac{\sum_{c\in S}\sum_i w_i^c z_i^c \delta(x_i^j \leq \theta)}{\sum_{c\in S}\sum_i w_i^c \delta(x_i^j \leq \theta)}, \tag{7}$$

$$k^c = \frac{\sum_i w_i^c z_i^c}{\sum_i w_i^c} \quad c \notin S. \tag{8}$$

Testing all possible subsets of classes is exponential in the number of classes. This makes a naive training implementation very slow. To speed up the process, we implement some of the modifications suggested in [19] but keep a complete search over the sharing combinations.

4. Experiments

For the experiments we used the Weizmann database compiled originally for [1]. This database presents 10 actions (bending, jumping jack, jumping, jumping in place, running, galloping sideways, skipping, walking, one-hand waving and two-hand waving) performed by nine individuals. It is a simple dataset containing only one person per video and no complex background.

For training we manually extract one keyframe and its surrounding frames from each video to estimate optic flow (Fig. 2). We then extract 2250 image fragments from this set. We adopt a leave-one-out cross validation scheme and an eight bin discretisation of the optic flow for all testings.

Prior to performing experiments with complete videos, we tested the difference between detection and classification performance using only shape information and shape plus motion. For these simpler experiments we tested frames selected to contain a keyframe. Training both classifiers using 100 stumps, the shape-only classifier had a poor overall classification accuracy of 37.5%, compared to the 92.9% of the shape-motion one. Adding 500 background negative examples (taken from random images) to the shape-only classifier helped to raise its performance up to 64.9% still far below the shape-motion classifier. Fig. 6 shows some of the results obtained with the shape-motion classifier.

To classify the entire video database, the following procedure is performed. Training is divided in two phases: first a preliminary joint classifier is trained using only positive keyframe examples of each class (the negative examples of

Figure 5. The handpicked keyframes chosen to represent the actions: bending, jumping jack, jumping, jumping in place, running, galloping sideways, skipping, walking, one-hand waving and two-hand waving.

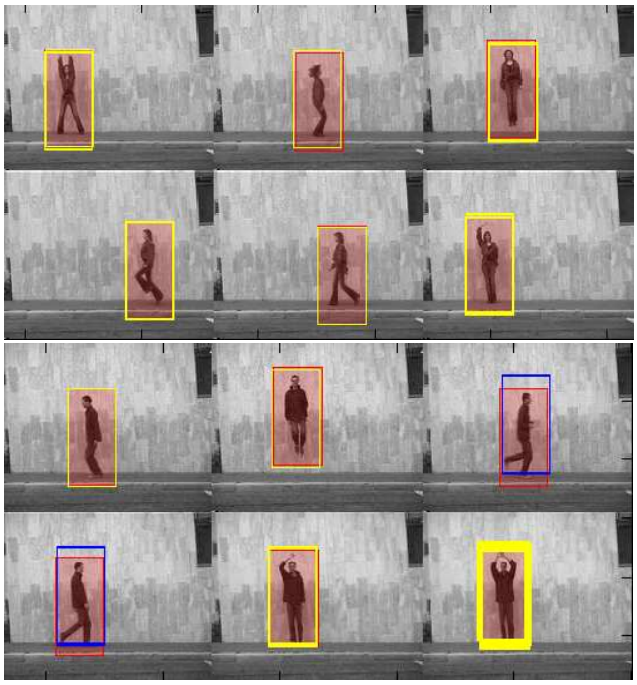

Figure 6. Testing results. Ground truth (red), correct classification and localisation (yellow), incorrect classification (blue).

one class are the positive examples of the other classes). We use this classifier to test the video database and collect false positives. Second, we retrain the classifier adding the false positives as negative examples. The final multiclass classifier is trained using 200 stumps (Fig. 4). For the testing part, we search the first 60 frames of each video for keyframes. Local maxima detections in a temporal window of 10 frames are selected. These selections vote for their class according to their classification magnitude. A frame-by-frame example of the classification magnitude of some tested videos is shown in Fig. 7. It can be seen that local maxima correspond to correct keyframe detections. Following this testing framework we achieved 100% percent of detection and classification accuracy. Comparing classification and localisation results with other works in the literature is difficult due to the very method-specific testing procedures developed; yet, to put our results in context we present a summary of the best classification results obtained using the Weizmann database in Table 1. This results consist of correct classification of whole video sequences. It is also worth to notice that the methods in [17] and [20] don't localise actions.

Method	Accuracy
Shared shape-motion parts	**100 %**
SCHINDLER [17]	**100 %**
BLANK [1]	**100 %**
JHUANG [8]	98.8 %
WANG [20]	97.8 %
NIEBLES [15]	72.8%

Table 1. Comparison of the classification accuracy for the Weizmann database of our method with the results obtained in the latest papers.

In order to get insight into the method's performance in complex scenes, we tested a classifier trained with the Weizmann database, with the videos used in [9]. In these videos five of the 10 learned actions are present (bending, jumping jack, walking, one-hand waving and two-hand waving). Examples of these videos can be seen in Fig. 1. Some results are shown in Fig. 8. These are divided in (1) correct detection and classification, (2) correct detection / wrong classification and (3) false positives. Many of the false positives were detected in videos with motion clutter. An explanation for this behaviour could be that the patches selected as part of the joint classifier during the training process didn't have enough discriminant power in motion cluttered scenes due to the lack of background motion in the training examples. Despite this disadvantage and the complexity of the videos, we were still able to detect several keyframes.

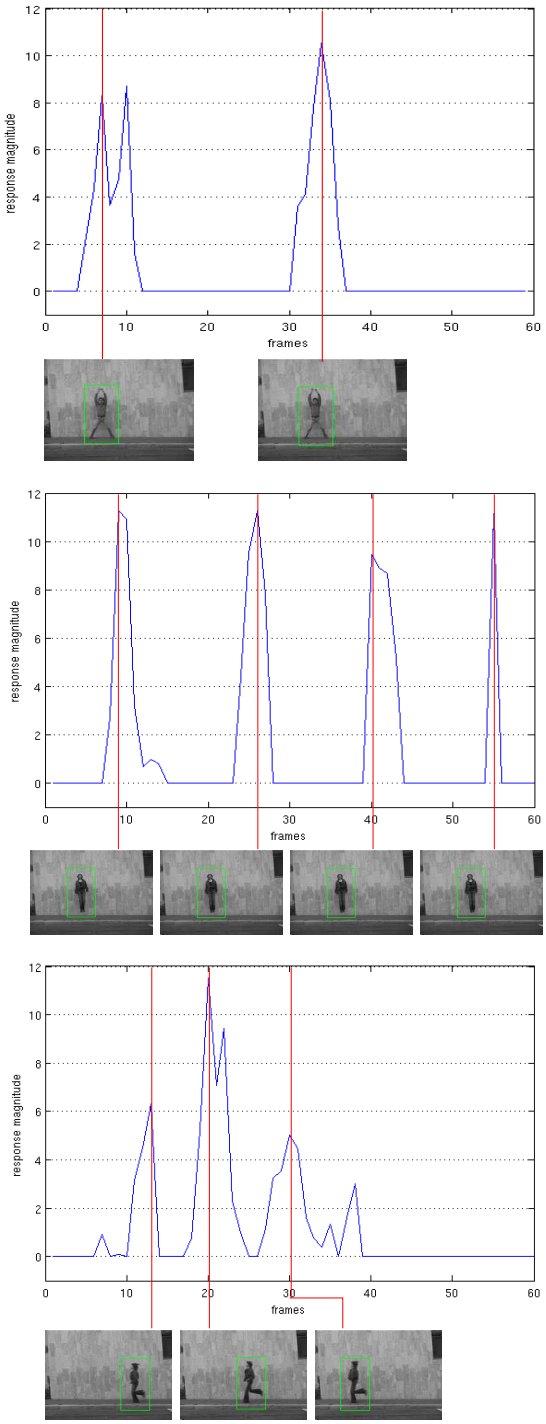

Figure 7. Maximum value of the classifier in each frame in some of the testing videos (of the actions jumping jack, jumping in place and running), as we can see, high detection values correspond to matching keyframes (Fig. 5).

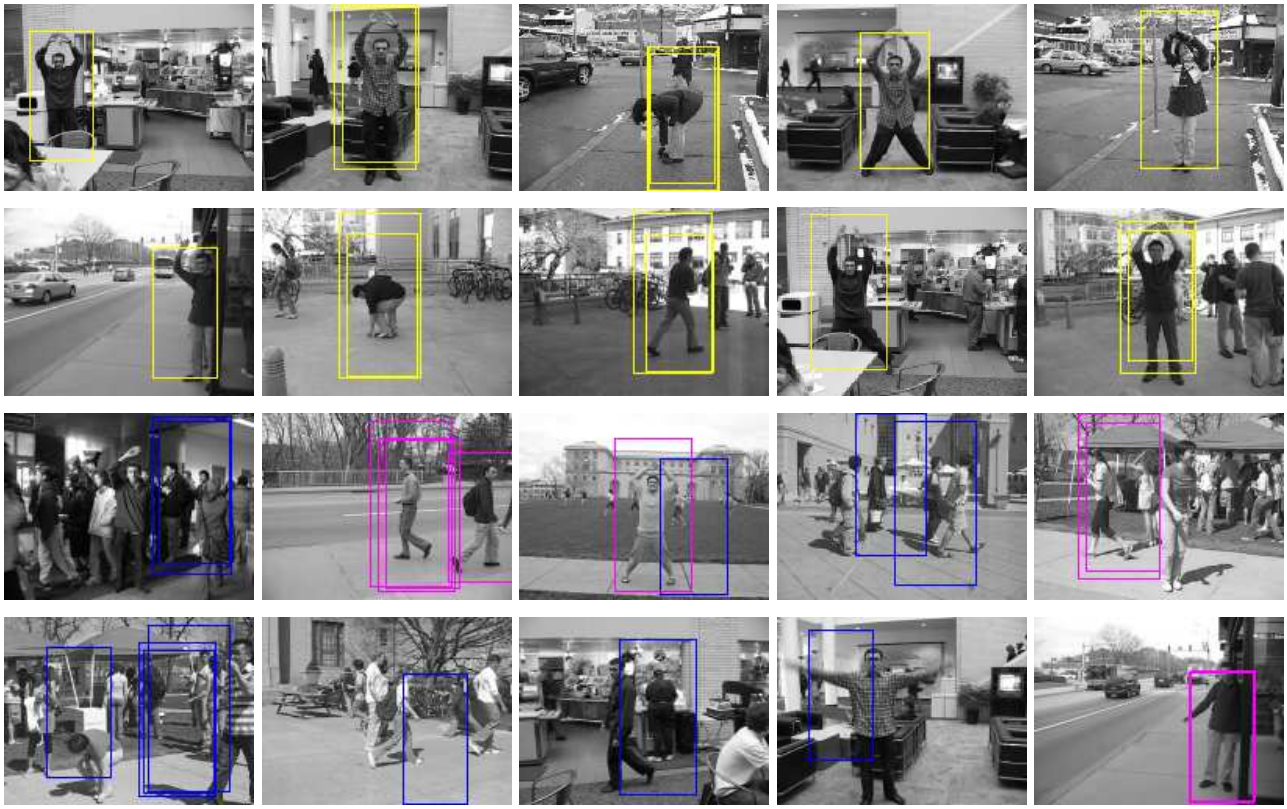

Figure 8. Tests with a 200 stump classifier in scenes with complex background. First two rows, examples of correct detection and classification (yellow). Last two rows, examples of misclassification (magenta) and false positives (blue).

5. Conclusions

In this paper we showed that a simple keyframe technique is enough to detect action in videos. We added motion information to a well-known boosting method, improving its accuracy for the task of human action detection and classification. We also showed promising results in complex scenes that could be improved by adding a proper set of training examples.

Because of the good results showed by other methods using the shame action database further comparisons are required to highlight the advantages of our method, namely, recognition time, number of parts used and performance as a function of the number of training examples. All these will be addressed as part of our future work.

Despite the obtained results, we noticed some shortcomings of the approach. Currently, we are using intensity-based fragments and NCC as a shape similarity measure; this has the disadvantage of lacking robustness when matching people wearing different clothes, relying on the amount of training examples to overcome this difficulty. Future work on this topic will include better shape and motion descriptors for the fragments, in particular histograms of oriented gradients (HoG) that have shown a good performance in [12].

Histograms of Optic Flow seem also like an obvious extension for adding robustness to the patch motion descriptor. Mantaining a more detailed record of all the flow directions inside the patch could help in selecting which directions are important for an action patch and wich are part of the background without having to rely in silhouette segmentations.

Other improvements include automatic keyframe selection and quantitative testings in long video sequences including complex and cluttered backgrounds.

6 Acknowledgements

We would like to thank Yan Ke for allowing us the use of his action database. This work was supported by CONACYT and the EU FP6 project *HERMES*.

References

[1] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as Space-Time Shapes. In *International Conference on Conputer Vision*, 2005.

[2] S. Carlsson and J. Sullivan. Action Recognition by Shape Matching to Key Frames. In *Workshop on Models versus Exemplars in Computer Vision*, 2001.

[3] N. Dalal, B. Triggs, and C. Schimd. Human Detection Using Oriented Histograms of Flow and Appearance. In *European Conference on Computer Vision*, 2006.

[4] J. Davis. Recognizing Movement using Motion Histograms. Technical Report 487, MIT Media Lab Perceptual Computing Group, MIT, 1999.

[5] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, pages 65–72, 2005.

[6] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing Action at a Distance. In *International Conference on Computer Vision*, Nice, France, 2003.

[7] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of statistics*, 28(2):337–374, 2000.

[8] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A Biologically Inspired System for Action Recognition. In *International Conference on Computer Vision*, 2007.

[9] Y. Ke, R. Sukthankar, and M. Herbert. Event Detection in Crowded Videos. In *International Conference on Computer Vision*, 2007.

[10] I. Laptev and T. Lindeberg. Space-time Interest Points. In *International Conference on Computer Vision*, 2003.

[11] I. Laptev, M. Marcin, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *Conference on Computer Vision and Pattern Recognition*, 2008.

[12] I. Laptev and P. Perez. Retrieving Actions in Movies. In *International Conference on Computer Vision*, 2007.

[13] B. Lucas and T. Kanade. An Iterative Image Registration Technique with an application to Stereo Vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[14] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words. In *British Machine Vision Conference*, 2006.

[15] J. C. Niebles and L. Fei-Fei. A Hierachical Model of Shape and Appearance for Human Action Classification. In *Conference on Computer Vision and Pattern Recognition*, 2007.

[16] A. Oikonomopoulos, P. Ioannis, and M. Pantic. Spatio-Temporal Salient Points for Visual Recognition of Human Actions. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 36(3):710–719, 2006.

[17] K. Schindler and L. Van Gool. Action Snippets: How many frames does human action recognition require? In *Conference on Computer Vision and Pattern Recognition*, 2008.

[18] J. Sullivan and S. Carlsson. Recognizing and Tracking Human Action. In *European Conference on Computer Vision*, 2002.

[19] A. Torralba, K. Murphy, and W. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[20] L. Wang and D. Suter. Recognizing Human Activities from Silhouettes: Motion Subspace and Factorial Discriminative Graphical Model. In *Conference on Computer Vision and Pattern Recognition*, 2007.

[21] Y. Wang, H. Jiang, M. Drew, Z. Li, and G. Mori. Unsupervised Discovery of Action Classes. In *Conference on Computer Vision and Pattern Recognition*, 2006.