A Video Self-descriptor based on Sparse Trajectory Clustering

Ana Mara de Oliveira Figueiredo[1], Marcelo Caniato[1], Virgínia Fernandes Mota[2], Rodrigo Luis de Souza Silva[1], and Marcelo Bernardes Vieira[1]*

[1] Universidade Federal de Juiz de Fora, Juiz de Fora, Brasil
{anamara,marcelo.caniato,rodrigoluis,marcelo.bernardes}@ice.ufjf.br
[2] Colégio Técnico, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil
virginiaferm@dcc.ufmg.br

Abstract. In order to describe the main movement of the video a new motion descriptor is proposed in this work. We combine two methods for estimating the motion between frames: block matching and brightness gradient of image. In this work we use a variable size block matching algorithm to extract displacement vectors as a motion information. The cross product between the block matching vector and the gradient is used to obtain the displacement vectors. These vectors are computed in a frame sequence, obtaining the block trajectory which contains the temporal information. The block matching vectors are also used to cluster the sparse trajectories according to their shape. The proposed method computes this information to obtain orientation tensors and to generate the final descriptor. The global tensor descriptor is evaluated by classification of KTH, UCF11 and Hollywood2 video datasets with a non-linear SVM classifier. Results indicate that our sparse trajectories method is competitive in comparison to the well known dense trajectories approach, using orientation tensors, besides requiring less computational effort.

Keywords: Block Matching, Human action recognition, Self-descriptor, Sparse and dense trajectories, Trajectory clustering

1 Introduction

Human action recognition is a challenging problem in Computer Vision, and it has been an active area of research for over a decade. An important part of the recognition process is the representation of motion information. This was approached in different ways over the course of the years. Recently, using video descriptors for representing this kind of information have become a trend among researchers. In this context, orientation tensors were used in [1–3] for describing video information. The tensor keeps data that represent the relationship between vectors coefficients associated with the original motion information. Likewise, our work also employs tensors. Our descriptor is the result of the concatenation

* The authors thank FAPEMIG, CAPES and UFJF for funding.

of separate cluster vectors obtained from cross products between two different motion informations.

Two different approaches are combined here in order to detect motion displacement. The first one is block matching. The other one is the calculation of the brightness gradient between two consecutive frames. The cross product between the vectors of these two methods is calculated, and then this resulting vector is clustered according to the mean angle of the block matching trajectory vectors. Following other works in the literature, we use spatial-temporal motion information for better representing the motion.

In order to evaluate the video descriptor obtained by the proposed method, we used well-known datasets, such as KTH, UCF11 and Hollywood2. Besides, a Support Vector Machine (SVM) was employed for classification purposes.

2 Related works

In [4], we have used a block matching method to extract motion information from a video. In this method, each frame is divided into blocks of a predetermined size, and each block is matched to a correspondent block in a subsequent frame. The matching can occur in a sequence of frames in order to obtain the trajectory of a block within the sequence. The method calculates displacement vectors for each block and uses a histogram to quantize these vectors. This histogram is also used to calculate a tensor capable of describing a video.

The idea of using trajectories for extracting human activities has become increasingly popular. A work that uses the space-time trajectories idea is presented in [5]. The motion is decomposed into dominant and residual parts. These parts are used in the extraction of space-time trajectories and for the descriptor computation. The resulting descriptor, named DCS, is based on differential motion scalar quantities, as well as on divergence, curl and shear features (hence the acronym DCS). It uses the Vector of Local Aggregated Descriptor (VLAD) encoding technique to perform action recognition.

The residual motion discussed above also plays an important role in the method proposed by [6]. In their work, two methods are combined to estimate camera motion: SURF descriptors and dense optical flow. Feature points are matched and used to estimate a homography with RANSAC. A human detector is also employed to remove inconsistent matches originated from human activity and trajectories generated from camera motion.

Finally, [1] presented a tensor motion descriptor for video sequences using optical flow and HOG3D information. In that work, they use an aggregation tensor-based technique, combining two descriptors. One of them carries polynomial coefficients which approximate optical flow, and the other one carries data from HOGs. This descriptor is evaluated by a SVM classifier using KTH, UCF11 and Hollywood2 datasets. Orientation tensors are also used as motion descriptors in [7] in conjunction with dense optical flow trajectories. For each optical flow point, he computes the cross product between the trajectory displacement vector and the 3D gradient vector in a window around that point. The opti-

cal flow displacement vectors are also used for clustering based on trajectories shape. According to this clustering, the resulting motion vectors are grouped and represented by a tensor.

In the new method presented in this work, block matching is employed as in [4], but a Variable Size Block Matching Algorithm (VSBMA) is used instead to obtain a better division of the frame. In our work, the approach of using block matching vectors considerably reduces the effort needed for tracking motion, as compared to the use of HOG3D and dense trajectories.

3 Proposed Method

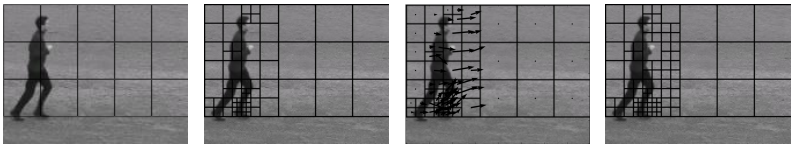
The basis for our proposed method will be presented in the following sections. The method presentation will be divided in five stages. In general terms, we calculate the block displacement using the 4SS block matching. The displacement vector is used in the computation of the cross product with the brightness gradient in the block area. The resulting three-dimensional vectors are then clustered according to the average of the angles of trajectory vectors.

Two different variations of the method were used during experiments. These differences are entirely concentrated in the first stage, presented in Section 3.1, where the sparse trajectories are computed using block matching. Both the method of block division and the trajectory formation are switched in each variation. The subsequent stages remain unchanged.

3.1 Computing sparse trajectories

The input of our method is a video, i.e., a set of frames $V = \{F_k\}$, where $k \in [1, n_\mathrm{f}]$ and n_f is the number of frames. The goal is to characterize a video sequence action, so the movement is described for $k-1$ frames, because at least one successor frame is needed for the calculation. We also need to calculate the displacement block between the first two frames of the sequence. For that purpose, a Variable Size Block Matching Algorithm (VSBMA) is used. The frame k of the video is subdivided into $n_x \times n_y$ non-overlapping blocks of exactly $s_0 \times s_0$ pixels, where s_0 is an initial block size fixed for the first frame. If the frame dimension is not a multiple of s_0, the remaining right and bottom pixels do not form blocks, as can be seen in Figure 1(a). The algorithm searches for each block from the reference frame in the target frame based on any BMA search strategy. If the match error found is greater than a fixed threshold, the block is split into four half-sized blocks until the error is below the threshold or it reaches a predetermined smallest permitted size s_s. A quadtree is used for this purpose with leaf nodes corresponding to blocks of varying sizes. The objective is to make the edge of the blocks coincide with the border of objects in the scene, forming regions with uniform displacement. After this block division, a final number of blocks n_b is reached, where $n_b \geq n_x \cdot n_y$, as can be seen in Figure 1(b).

VSBMA is used here with a 4SS strategy to generate the displacement vectors map. For each block B_i, displacement vectors $\boldsymbol{v}_i{}^k = (x, y) \in \mathbb{R}^2$ are calculated,



(a) 3x5 blocks of 32x32 pixels (b) 69 blocks of variable sizes (c) VSBMA displacement vectors map (d) An example of a block division in the second variation

Fig. 1. An example of motion estimation using VSBMA

where $i \in [1, n_b]$ is the block index. In this way, the most representative size is selected for each region of the frame. Specifically, the frame is divided into blocks with a predetermined initial size, which are split only when the lowest Block Distortion Measure (BDM) is still above the established threshold. Thus, a single frame can have blocks of various sizes, and their displacement vectors are all in the same map.

In order to explore the information of object trajectory through the video, we use t pairs of adjacent frames from the sequence, where t is the size of the trajectory generated from $t + 1$ frames. The block in the reference frame k is matched with a block in the 'target' frame, finding the correspondent block in frame $k + 1$. The correspondent block found is used as a reference block for the next match. The subsequent pairs use the same block size configuration as the first pair in the sequence.

An initial frame k and t successor frames are used to generate t vectors for each block trajectory, starting on the original grid. The number of frames used in the trajectories are predetermined and equal for all video.

The vector describing the displacement of a block between two consecutive frames is defined as $\boldsymbol{v}_{i,j}^{k} = (x, y) \in \mathbb{R}^2$, where k is the index of the initial frame of the sequence and $j \in [1, t]$ is the index of the reference frame used to compute the vector. The set of these t vectors of all blocks form a map of motion vectors referent to k-th frame, as can be seen in Figure 1(c), where $t = 4$.

3.2 Generating histograms using cross product vectors

The same reference frames used to compute the BM are selected as references to compute the brightness gradient. These brightness gradient vectors are used to calculate the cross product with BM displacement vectors. As these displacement vectors have only the spatial components (x, y), a third component representing the time displacement is added, in order to make possible the cross product computation. As only a single frame displacement is considered, the temporal coordinate is set to 1.

These calculated cross products are represented by a vector set defined as:

$$C_{i,j}^{k} = \{\boldsymbol{c}_q^{\,k} \mid q \in B_{i,j}^{k},\ \boldsymbol{c}_q^{\,k} = \boldsymbol{v}_{i,j}^{k} \times \boldsymbol{g}_q^{\,k}\}, \tag{1}$$

where q is a point in the block $B_{i,j}^k$ and $\boldsymbol{g}_q{}^k$ is the brightness gradient in that point.

The motion vector set obtained from each block is represented by a motion estimation histogram. All the resulting vectors in $C_{i,j}^k$ are converted to equivalent spherical coordinates and quantized into histogram $\boldsymbol{h}_{i,j}^k$. Then, we normalize these histograms, as is common in computer vision, to make it invariant to brightness magnitude. We have tested the L^1 and L^2 norm for histogram normalization and the first one obtained better results.

3.3 Clustering the trajectories of a frame

In order to address the problem of camera motion, trajectories are clustered based on their shape. As this clustering groups trajectories with similar angle sequences, the camera motion trajectories tend to be in the same cluster.

For each trajectory an angle vector is associated, which has a length equal to $t-1$. The angles are calculated between two consecutive displacement vectors of the trajectory. The vector $\boldsymbol{a}_i{}^k$ of the trajectory i is created using $\boldsymbol{a}_i{}^k = (a_{i,1}^k, ..., a_{i,t-1}^k)$, as the following equation

$$a_{i,j}^k = \cos^{-1}\left(\frac{\boldsymbol{v}_{i,j}^k \cdot \boldsymbol{v}_{i,j+1}^k}{\parallel \boldsymbol{v}_{i,j}^k \parallel \cdot \parallel \boldsymbol{v}_{i,j+1}^k \parallel}\right), \tag{2}$$

where $j \in [1, t-1]$ is the index of the vector element.

The result of this equation is a number between 0 and 180, because it gives us the smallest angle between two connected vectors in the trajectory. This angle vector is then used in k-means clustering, and the mean of all angles of each cluster is used to sort the cluster.

The number of clusters n_c is predetermined and stay the same for the whole video. Each cluster $X_c = \{i \mid \boldsymbol{a}_i{}^k$ was assigned to the cluster $c\}$ tends to have very similar trajectories.

3.4 Generating the frame descriptor

The histograms, $\boldsymbol{h}_{i,j}^k$, are represented by an orientation tensor $\mathbf{T}_{i,j}^k = \boldsymbol{h}_{i,j}^k \cdot \boldsymbol{h}_{i,j}^{k\,T}$ where $\mathbf{T}_{i,j}^k \in R^{n_\theta}$.

Individually, these tensors have the same information as $\boldsymbol{h}_{i,j}^k$, but several tensors can be combined to find component correlations. In order to determine an orientation tensor that describes the block displacement in the i-th trajectory we sum all tensors along the same trajectory into a single tensor $\mathbf{T}_i^k$:

$$\mathbf{T}_i^k = \sum_{j=0}^{t} \mathbf{T}_{i,j}^k. \tag{3}$$

For the frame k, we generate c tensors $\mathbf{T}_c^k$, which together contains all information about the frame trajectories. These tensors are composed by tensors

corresponding to trajectories associated to this cluster position c:

$$\mathbf{T}_c^k = \sum_{i \in X_c} \mathbf{T}_i^k. \tag{4}$$

As proposed by [7], the clusters c of each frame are ordered using the ascending angles ϕ and θ. The tensors $\mathbf{T}_c^k$ follow the same ordering. It ensure that the vectors with similar angles tend to be in the same cluster number for all frames.

3.5 Generating the final video descriptor

At this point, all frames have c tensors in ascending order. In order to obtain the information about the whole video, we accumulate the tensors of all frames to obtain the descriptor for the video. To make it possible, for each cluster c, we generate a tensor $\mathbf{T}_c$ for the video by accumulating the tensors $\mathbf{T}_c^k$:

$$\mathbf{T}_c = \sum_{k=0}^{n_\mathrm{f}} \mathbf{T}_c^k, \tag{5}$$

respecting the ascending order given by the histogram angles. In other words, the tensors corresponding to smaller angles of all frames will be added together, the tensors corresponding to second smaller angles of all frames will be added together and so on.

Finally, we concatenate the tensors $\mathbf{T}_c$ to obtain a vector $\boldsymbol{d}$ that describes the video:

$$\boldsymbol{d} = (T_1^k, ..., T_c^k)$$

This descriptor encapsulates all information about the motion within the video.

3.6 Variations of the method

In the first variation of the proposed method the process of block division occurs between the first two frames of the sequence. The second variation modifies the block division process, by allowing it to occur in every frame of the sequence.

The first variation is a result of the steps outlined in the previous sections. First, VSBMA displacement vectors are computed using t pairs of adjacent frames, as shown in Section 3.1. In this phase, we use the pair of frames t_0 in the block division process, and this final division defines the size of each block. This size remains the same for the processing in the next frames.

In the second variation, a modification is introduced in the block division part of VSBMA. We use all trajectory frames in this division. We initially divide frame f_0 into blocks of a predetermined initial size. Each block is searched in the subsequent frame f_1 and can be divided according to the match error, as explained in Section 3.1. In the first variation, we had two conditions for ending the division of a block: the match error should be smaller than the threshold or

the division should result in four blocks smaller than the smallest predetermined size. In this second variation, if we reach the first condition, we try to continue the process using the next frame f_2. The division process proceeds until the block size reaches the smallest size. In Figure 1(d), we can notice that the number of blocks increased considerably compared to the previous method. This happens because all frames contribute to the division process, and frames in the end of the sequence produce additional divisions when compared to the other variations.

4 Experimental Results

Our proposed method makes use of six parameters. Three of them are important for VSBMA: the initial block size, the minimum allowed size for blocks and the threshold of block division. These parameters directly affect the final division of the blocks and, consequently, the descriptor performance. Plus, we have the trajectory size t, which is the amount of frame pairs used in the computation of displacement vectors. This number has to be large enough to satisfactorily catch the temporal information. The number of bins is another parameter. It affects the histogram, since this is uniformly divided into a specified number of bins. Finally, the number of clusters c is used by the k-means algorithm for grouping samples. All clustering steps are done using the stopping criterion of 500 iterations or less than 0.01% of change in the cluster membership from last iteration.

The main goal of this experimentation is to obtain the best parameter configurations, in order to compare our results with similar works. The proposed method was then tested with different values for each parameter in the KTH dataset. In order to test our descriptor in more realistic scenarios, we also used the UCF11 and Hollywood2 datasets, which have more action classes and colorful videos. We performed these tests using the best parameter configuration found in KTH tests. As the videos in these datasets require more computational resources, varying the parameters would not be viable.

In order to measure the recognition rate achieved using our descriptors, we use an SVM classifier, which takes the descriptors for the whole database. We follow the same classification protocol as [8]. We use both Triangle and Gaussian kernel and a one-against-all strategy for multiclass classification. We produce 6 recognition rates for each parameter combination, 3 using a triangular kernel, and 3 using a Gaussian kernel. In the following sections, we present the highest result achieved from these ones.

4.1 Results for the first variation

In order to determine the best parameter configuration for our tests, we decided to run experiments by incrementally fixing each parameter, based on empirical knowledge acquired during the development of this work. We started by evaluating the effect of varying the first parameter, i.e., the initial block size. By analyzing the results, we fixed this parameter with the size value that produced

the best recognition rate. Then, we moved on to the second parameter and initiated a new set of experiments to determine, out of a set of predetermined test values, which one yielded the best result. As before, once tests were finished, the best value was permanently assigned to the parameter, initiating, then, tests for determining a value for the third parameter, and so on. This process continued until we were able to fix all six parameters. This set of values formed our intended best parameter configuration. Below we present the set of values tested for each parameter:

- Initial block size: 16×16, $\mathbf{32 \times 32}$ and 48×48
- Minimum allowed block size: $\mathbf{4 \times 4}$ and 8×8
- Threshold: 5, 7, **10**, 15, 20, 25 and 30
- Number of bins: 12, 24, **26** and 30
- Number of clusters: 1, 2, **3** and 4
- Trajectory size: 2, 4, **6**, 8 and 10

The values highlighted in the list above are the ones that compose the final configuration. Figure 2 shows the results obtained for each parameter variation. Notice that the trajectory size was also varied during each experiment. In the majority of tests, the best result were achieved with a trajectory size of 6. We tried to find a combination of parameters that would provide good results with a satisfactory number of trajectories. Ideally, this number has to be small enough so as to not require too much computing and large enough to accommodate important information.

Although we empirically determined the best values for each parameter, we can interpret the results to provide some insights on what might actually be happening behind the scenes. For example, during the initial block size tests, we noticed a significant increase in recognition rate when a 32×32 size was used. This is depicted by the red curve in Figure 2(a). We believe this is related to having some blocks in the region without major movements when their size is 16×16, besides describing redundant information. We can also observe the effects of varying the smallest permitted block size, shown in Figure 2(b). One can observe that the rates for 8×8 size (shown in blue) tend to worsen as the trajectory size increases. Besides, its best result is still much below the 4×4 curve. It occurs because blocks of smaller sizes can correctly represent regions of important movement.

In Figure 2(c), we can note the threshold of 5 is associated with the worst curve of the graph. This value causes the formation of many trajectories, and these may have noisy information. The best results of these tests was obtained for two threshold values: 10 and 25. A threshold of 10 still causes the generation of many trajectories, consequently worsening performance. However, considering a larger span of trajectory sizes, a threshold of 10 could maintain a better rate than when using 25. This is the main reason why the value of 10 was chosen for the subsequent tests.

Figure 2(d) shows the influence of the number of bins in the results. 30 and 26 bins both achieved good recognition rates. Although experiments with 30 bins

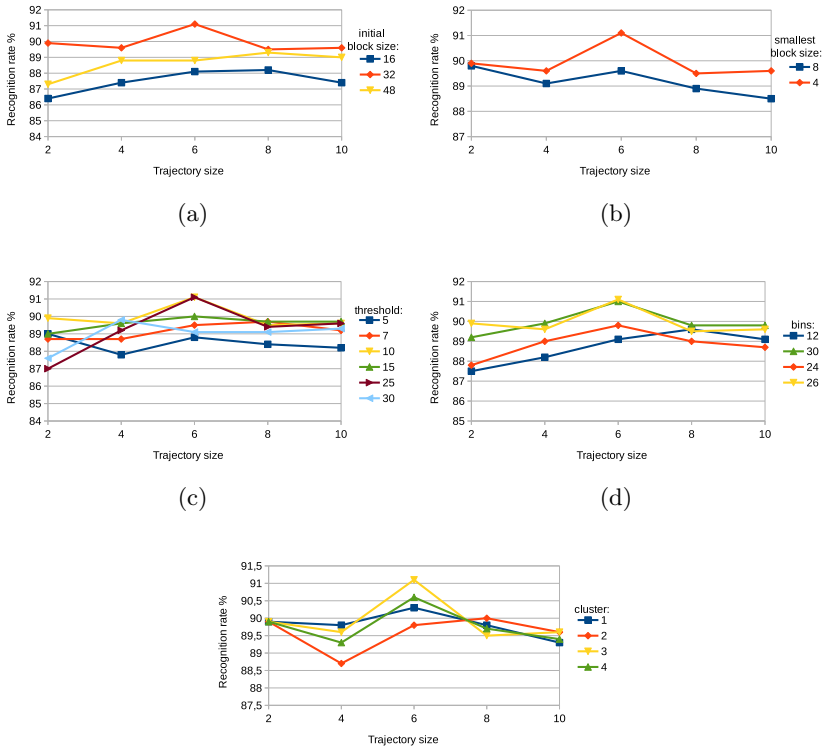


Fig. 2. Experiments on KTH dataset with first variation

achieved better results than those with 26 bins for trajectories of size 4, 8 and 10, we decided to use 26 bins in the subsequent tests, mainly because its result was better when 6 frames were set as the trajectory size. Small values of bins implies to larger ranges of angles and, consequently, groupings of vectors with different information. On the other hand, big values can generate empty bins, increasing the descriptor size with no information gain. With respect to the number of clusters used in the k-means algorithm (Figure 2(e)), we use 1 cluster in order to show that the absence of trajectory clustering results in a worse recognition rate. If we observe the results for the trajectory size equal to 6, we can see that, as the number of clusters is incremented, the recognition rate only increases until a certain point, where it begins to fall. This shows a tendency for the best number of clusters for this dataset to be 3. As our method uses sparse trajectories, we achieve better results using a small number of clusters, in contrast to [7], which uses 5 clusters.

The tests above resulted in a recognition rate of 91.1% for the KTH database, using the best parameter combination mentioned earlier. We also performed tests in UCF11 and Hollywood2 datasets using the best parameters found for KTH. We achieved 65.8% of recognition rate in UCF11, where simple actions, such as *jumping, diving, riding*, have recognition rates above 80%. On the other hand, more complex actions were mistaken for others, as *juggling* for *jumping*, *shooting* for *tennis*, and *walking* for *riding*. The average recognition rate was 41.5% in Hollywood2 tests. Actions with the main movement in small regions, as answer phone and sit up, present the worst results for this dataset.

4.2 Results for the second variation

In order to perform tests with the second variation using KTH, we varied the parameters values as in previous experiments. We initialize the tests using 32×32 as the initial block size, smallest block size of 4×4, 26 histograms bins, threshold of 10 for VSBMA, 3 clusters in K-means algorithm and 2, 4, 6, 8 and 10 frames for the trajectory size.

Varying the initial block size, the test which achieved the best result was the (32×32) one, where we obtained 89.9% of recognition rate, using 2 as the trajectory size. Increasing the trajectory size does not improve results when using this parameter configuration. Testing the smallest block size we have better results using (4×4). Its tests shows better results than (8×8) for all trajectory sizes. This behavior has already been observed in previous test, shown in Figure 2(b).

In the threshold tests, we achieved our best results by using 20 and 25 as thresholds, yielding 91.1% and 91.4% of recognition rate for trajectory sizes of 4 and 6, respectively. As the 25 tests shown a better behavior, besides decreasing computational cost, this value was chosen for subsequent tests. The threshold value directly influences the number of trajectories: the higher the value, less trajectories are formed. As this variation form more trajectories than the previous one, the threshold 25 maintains a reasonable amount of trajectories, despite being a high value.

The tests for 26 bins present the best results for all trajectory sizes tested. It achieved 91.4% of recognition rate using a trajectory of size 6. This was also the best rate obtained for number of clusters tests. These experiments showed a behaviors similar to the ones observed in the first variation tests, shown in Figure 2(d) and 2(e). Once again, we observed that 6 is an appropriate value for the trajectory size in this dataset.

In Table 1, we show the confusion matrix of our best result in KTH. The main diagonal of this matrix represents the percentage of correctly classified videos. In this test, we achieved a recognition rate of 91.4% using 32×32 as initial block size, 4×4 as smallest block size, 25 as threshold, 26 bins, 3 clusters and 6 as trajectory size. In *boxing*, *clapping*, *waving* and *running* classes, we achieved better recognition than using our previous method variations. Our video descriptor recognized the *boxing* action in 100% of boxing-videos. However, we had a small worsening in recognition for *jogging* and *walking* classes, comparing

with first variation results. These are the two more complicated actions of this dataset.

Table 1. Confusion matrix of the best result on KTH dataset with third variation. The average recognition rate is 91.4%.

	Box	HClap	HWav	Jog	Run	Walk
Box	100	0.0	0.0	0.0	0.0	0.0
HClap	5.6	93.1	1.4	0.0	0.0	0.0
HWav	0.7	2.8	96.5	0.0	0.0	0.0
Jog	0.0	0.0	0.0	83.3	5.6	11.1
Run	0.0	0.0	0.0	21.5	76.4	2.1
Walk	0.0	0.0	0.0	0.7	0.0	99.3

Concluding our experiments, we tested our best parameter configuration for this second variation against UCF11 and Hollywood2 datasets. For the UCF11 dataset, we achieved 65.6% of recognition rate. In the Hollywood2 dataset experiment we were able to achieve a recognition rate of 40.5%. This dataset has more complex videos than KTH. Hollywood2 videos usually present more than one person. The movement of these secondary persons cause wrong matchings, with bad impact in long trajectories.

4.3 Comparison with state-of-the-art

In Table 2, we show a comparison of our best result with state-of-the-art methods. Although it is clear that our results can not reach the state-of-the-art, that was not our initial goal. Our focus was on developing a method capable of obtaining reasonable results using sparse trajectories, which are computationally less expensive than dense trajectories.

Table 2. Comparison with state-of-the-art for KTH, UCF11 and Hollywood2 datasets.

	KTH	**UCF11**	**Hollywood2**
Klaser et al. [9] (2008)	91.0		24.7
Perez at al. [3] (2012)	92.0		34
Mota et al. [1](2013)	93.2	72.7	40.3
Wang et al. [6] (2013)	**95.3**	**89.9**	59.9
Wang and Schmid [10](2013)			**64.3**
Figueiredo et al. [4] (2014)	87.7	59.5	34.9
Caetano [7](2014)	94.1		46.5
Our method	**91.4**	**65.8**	**41.5**

In that matter, we were able to obtain satisfactory results. Even though we did not achieve recognition rates as high as those of dense trajectories, our method uses considerably fewer trajectories and, therefore, requires less computational effort. Despite rates being a bit lower, our results are very close to those obtained by other self-descriptor methods.

It is important to emphasize that most of these methods shown in Table 2 are not restricted to the self-descriptor constraint. Their video final descriptor are calculated using information of all videos of the dataset. Our descriptor, on the other hand, does not have any dependency on other videos. So, comparing our results with other self-descriptor methods, our results are better than those of [1] and [3] in the Hollywood2 dataset, which is, as mentioned before, the most complex set of videos used for testing in several works.

Some state-of-the-art methods shown in Table 2 present rates of 92% and above in KTH dataset, while our have shown a rate of 91.4%. Their results are very close to the limit of this dataset. When the recognition rate in a database reaches that point, it is very difficult to improve it further even by some tenths. Comparing our results against our first self-descriptor method, presented in [4], this method yields better results.

We perform some tests in order to estimate the computational effort of our method. We use a machine with Intel Xeon E5-4607, 2.20GHz, 32 GB of RAM using only 1 thread. We select 10 videos per dataset to perform this test quickly. The best parameter configuration previously found was used and our descriptors were computed with an average of 8.48, 1.66, 0.72 frames per second for the KTH, UCF11 and Hollywood2 videos, respectively. Tests in the same conditions were performed using the [7] method, which were computed with an average of 1.57, 0.61 frames per second for the KTH and Hollywood2 videos. This work does not provide results for the UCF11 database. As our method had higher processing rate, we can assume that our method requires lower computational cost than the dense trajectory method presented in [7].

5 Conclusion

This work presented a method for generating video descriptors based on sparse trajectories. Our descriptor is considered a self-descriptor, since it depends solely on the input video. It is computed by extracting and accumulating information from each frame of the video. Basically, the frame is divided into blocks and their displacement vectors are computed. These vectors are represented by a histogram, which is represented by a tensor.

We presented two variants of our method in this work. The first one uses only the first two frames of a sequence to perform the block division. After this division, the size of each block remains the same for the other frames of a sequence. This first variation achieved 91.1%, 65.8% and 41.5% of recognition rate in KTH, UCF11 and Hollywood2 datasets, respectively. In our second variation the block division was allowed in all frames of the sequence. Our best result in KTH dataset was achieved using this method variation, producing a recognition

rate of 91.4%. In UCF11 and Hollywood2 datasets we achieved 65.6% and 40.5% of recognition rate.

This work shows that it is possible to achieve good recognition rates using sparse trajectories. We use a smaller number of trajectories compared to the method of dense trajectories. Due to this characteristic, our data storing needs are reduced. It is important to note, though, that this amount of data is enough to represent the motion of the video.

References

1. Mota, Virginia F and Souza, Jéssica IC and Araújo, Arnaldo de A and Vieira, Marcelo Bernardes.: Combining Orientation Tensors for Human Action Recognition. In: Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 328–333, IEEE, (2013)
2. Sad, Dhiego and Mota, Virginia Fernandes and Maciel, Luiz Maurílio and Vieira, Marcelo Bernardes and Araújo, Arnaldo de A.:A Tensor Motion Descriptor Based on Multiple Gradient Estimators. In:Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 70–74, IEEE, (2013)
3. Perez, Eder de Almeida and Mota, Virgínia Fernandes and Maciel, Luiz Maurílio and Sad, Dhiego and Vieira, Marcelo Bernardes.: Combining Gradient Histograms Using Orientation Tensors for Human Action Recognition. In:21st International Conference on Pattern Recognition (ICPR), pp. 3460–3463, IEEE, (2012)
4. Figueiredo, Ana MO and Maia, Helena A and Oliveira, Fábio LM and Mota, Virgínia F and Vieira, Marcelo Bernardes.: A Video Tensor Self-descriptor Based on Block Matching. In:Computational Science and Its Applications–ICCSA 2014, pp. 401–414, Springer, (2014)
5. Jain, Mihir and Jégou, Hervé and Bouthemy, Patrick.: Better exploiting motion for better action recognition. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 2555–2562, IEEE, (2013)
6. Wang, Heng and Schmid, Cordelia and others.: Action recognition with improved trajectories. In: International Conference on Computer Vision, (2013)
7. Felipe Andrade Caetano.: A Video Descriptor using Orientation Tensors and Shape-based Trajectory Clustering. In: UNIVERSIDADE FEDERAL DE JUIZ DE FORA, (2014)
8. Schuldt, Christian and Laptev, Ivan and Caputo, Barbara.:Recognizing Human Actions: a Local SVM Approach. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), pp. 32–36, IEEE, (2004)
9. Alexander Kläser and Marcin Marszałek and Cordelia Schmid.: A Spatio-Temporal Descriptor Based on 3D-Gradients. In: British Machine Vision Conference (BMVC), pp. 995–1004, (2008)
10. Wang, Chunyu and Wang, Yizhou and Yuille, Alan L.:An approach to pose-based action recognition. In:Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pp. 915–922, IEEE, (2013)