International Journal of Image and Graphics

A Video Descriptor using Orientation Tensors and Shape-based Trajectory Clustering

Felipe Andrade Caetano
Organizational Knowledge Management Center, Universidade Federal de Juiz de Fora, Rua José Lourenço Kelmer Juiz de Fora, Minas Gerais, 36036-900, Brazil
felipe.caetano@ufjf.edu.br

Marcelo Bernardes Vieira
Department of Computer Science, Universidade Federal de Juiz de Fora, Rua José Lourenço Kelmer Juiz de Fora, Minas Gerais, 36036-900, Brazil
marcelo.bernardes@ice.ufjf.br

Rodrigo Luis de Souza da Silva
Department of Computer Science, Universidade Federal de Juiz de Fora, Rua José Lourenço Kelmer Juiz de Fora, Minas Gerais, 36036-900, Brazil
rodrigoluis@ice.ufjf.br

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Dense trajectories have been shown as a very promising method in the human action recognition field. In this paper, we propose a new kind of video descriptor, generated from the relationship between the trajectory's optical flow with the gradient field in its neighborhood. Orientation tensors are used to accumulate relevant information over the video, representing the tendency of direction in the descriptor space for that kind of movement. Furthermore, a method to cluster trajectories using their shape is proposed. This method allows us to accumulate different motion patterns in different tensors and easier distinguish trajectories that are created by real movements from the trajectories created by the camera's movement. The proposed method is capable to achieve the best known recognition rates for methods based on the self-descriptor constraint in popular datasets — Hollywood2 (up to 46%) and KTH (up to 94%).

Keywords: Dense Trajectory Clustering; Orientation tensor; Video Self-descriptor; Human action recognition.

1. Introduction

The human action recognition is a challenging problem in computer vision, which aims to automatically label which action is being performed by a human in a given

video. Due to the wideness of its potential applications, this task has been drawing a lot of attention over the last two decades.

Most of the main methods to address this problem is focused in extracting several small "visual words" (also called features) from the video and merging all the information together using machine learning. This approach is inherited from the image classification problem. Even some of the most common features extractor methods can be seen as an extension of the image case [1,2,3,4]. This kind of descriptor generalizes the problem of finding interesting features in the space to the space-time coordinates.

Among other space-time strategies, dense trajectories [5] and its related works [6,7,8] have shown the greatest recognition rates in the area, therefore being one of the most promising methods to address this problem. The dense trajectories method works by extracting stable points of each frame and tracking them in the subsequent frames, using an optical flow algorithm. A trajectory is created by connecting each one of these displacement vectors. Many features like the histogram of oriented gradients (HOG)[9], histogram of optical flow (HOF) [10] and motion boundaries histogram (MBH) [11] are extracted along the trajectory.

Although the process of gathering several small visual words using machine learning seems to achieve a very good recognition rate in most works, we believe this is not a requirement to achieve a successful recognition rate. In such a scenario, the final descriptor, computed for each video, is based on information that is assembled using all features from all videos in the training database. We believe it is possible to achieve good recognition rates using only the information provided by the video itself — namely self-descriptor. Self-descriptor is a constraint introduced by Figueiredo *et al.* [12] about how the final descriptor is generated for the video. As opposed to dictionary-based algorithms for human action recognition, a self-descriptor of a video is the final descriptor computed using exclusively the information provided by the video itself. No outside information can be used in the process of calculating the video's descriptor. A comparison between those two methods is shown in Figure 1.

This kind of approach might be good in many different situations. For example, when there are not enough videos to represent a particular action in the training database or the videos available for those actions does not generalize enough the action performed. Methods that are not based in the self-descriptor approach might overfit the training data in such situations. Additionally, methods using BoF and similar approaches do not admit well the insertion of new videos. In such case, all descriptors from all videos must be recalculated to admit the new video.

Another advantage of using the self-descriptor approach is that it requires much less space in both primary and secondary memory. This memory problem in the BoF often happens because all visual words from all the videos must to be stored before the clustering. A giant amount of space might be necessary. There is no such problem for the self-descriptor approach as the only space required is the one used by the final descriptor.

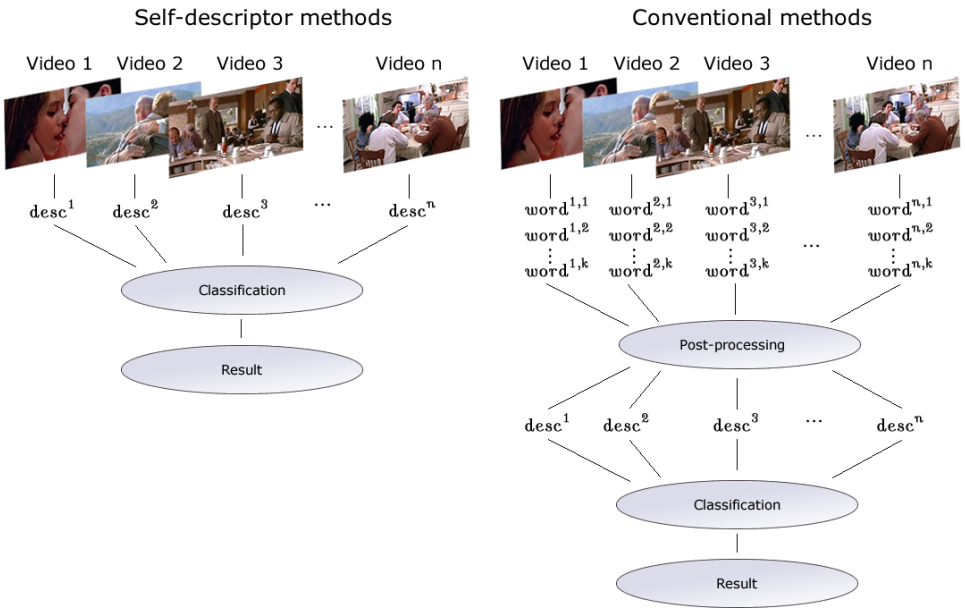

Figure 1. Comparison between Self-descriptor methods and conventional dictionary-based methods.

Our method has two main contributions. The first one is a method for combining trajectories, which represent interframe long-range motion, with gradients, which capture local brightness variation. This is achieved by a cross product between the local trajectory displacement vector and all the gradients vectors in a surrounding window. The resulting vectors are used to compute a local Histogram of Trajectory-Gradient Cross product (HTGC). The HTGC is capable of encoding both motion and shape information in a single representation. The second main contribution is a strategy to cluster trajectories based on their shape. We create a vector for each trajectory that represents its shape regardless its direction. By applying k-means to cluster those vectors, we aggregate trajectories that represent the same type of movement in the same cluster. Later, the proposed descriptor is used to address the human action recognition problem.

2. Related Works

2.1. *Tensor based approaches*

Tensors are extensions of the vector and matrix concept to higher orders. They have the power to describe relations between vectors, scalars, and other tensors and they are independent of a coordinate system. A special type of tensor is the orientation tensor which is a symmetric, positive-definite matrix. A vector $\vec{v}$ may be represented as an orientation tensor using $\mathbf{T} = \vec{v}\vec{v}^{\intercal}$. While vectors keep information

of one direction, orientation tensors might keep the tendency of many directions. In Figure 2 we can see graphical representation[13] of the different shapes that a 3-dimensional second-order tensor can have. The same idea is extended to higher dimensions. Tensors are a very powerful and robust tool that have been used in the last decade in many different works in the pattern recognition domain. This section presents works that use tensors to gather large amounts of data, storing most of the relevant information and the relationship between them as well.

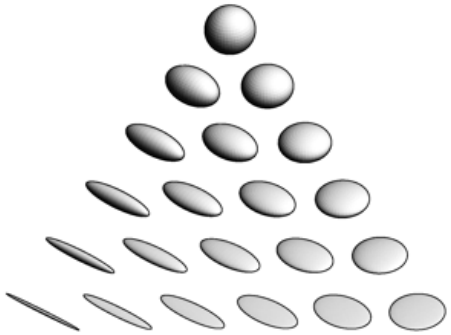

Figure 2. Graphical representation of 3-dimensional tensors.

Yuan *et al.* [14] created a new descriptor based on the covariance matrix of three concatenated low-level features: the position of the pixel in space and time, the gradient vector and the optical flow vector. These features are extracted on a spatio-temporal cuboid selected using the method proposed by Dollar *et al.*[15]. The authors argue that covariance matrices do not lie on Euclidean space, thus they apply a Log-Euclidean Riemann metric to measure the distance between those visual words and later they are quantized to form an appearance dictionary, as in BoF (Bag-of-Features) approach. A similarity between this work and ours is that we also use a covariance matrix to encode our descriptor. However, we not only use it just to represent the correlation between the coefficients of the original vector, but as a tensor that accumulates information along the whole video. Another similarity is that the covariance matrix in their proposed method is generated from a concatenation between vectors that holds different source of information. We also use this idea as we concatenate a histogram with a position point (x, y, t) before generating the tensor.

Perez *et al.* [16] proposed a method to address the human action recognition problem combining HOG3D [2] and orientation tensors. Each frame of the video is divided into equally sized windows and a tensor is computed for each one of the windows. After that, the tensor of the frame is computed by the sum of the normalized tensors of each window. The final descriptor is the sum of all frame's

tensor. The method shows a fairly competitive recognition rate, in despite of using a global approach (no Interest Points involved — all pixels are used to generate the final descriptor). Just like our method, the final descriptor in his approach is also calculated using only data provided by the video itself.

Mota *et al.* [17] also used orientation tensors to encode a descriptor based on HOG3D. But the author argues that aggregating several tensors naively over the time could lead to an isotropic tensor, which is a tensor that does not capture any main orientation and is thus useless for video description. Thus, they propose to concatenate multiple tensors in order to reduce overall isotropy. In their approach, the video is split in equally sized blocks and the orientation tensor computed over the HOG3D is calculated for each block. The final descriptor is the result of the concatenation of those blocks.

Mota *et al.* [18] proposed a method to address the human recognition problem combining what is proposed by Mota *et al.* [19] and Perez *et al.* [16]. The final descriptor is the concatenation of the tensors generated by both methods. The author shows that aggregating different source of information provided by the HOG and the optical flow descriptors may enhance the recognition rates in most of the popular databases.

Picard and Gosselin [20] improved the approach presented by Jegou *et al.* [21] by representing the descriptors as orientation tensors and aggregating them together around the same center. Although their work is focused in the content based image retrieval problem, this is still a good example of how using tensors to accumulate information can outperform other approaches based on vectors.

2.2. *Trajectory based approaches*

The dense trajectories proposed by Wang *et al.* [5] appeared as a good method to discriminate local motion information in videos and led to a wide collection of inspired works.

Jiang *et al.* [6] used k-means to cluster trajectories using the displacement between the first and last point of the trajectory. This is performed at each five frames and the trajectories are grouped into five clusters. The author assumed that the top three largest clusters are candidates to the dominant motion and use the mean motion of these to compensate other trajectories. After that, the author uses a pairwise motion representation to generate a visual codebook to the standard BoF approach. The similarity between this approach and ours is that we also try to cluster trajectories to find the dominant motion. However, we do not use the displacement of the trajectory and neither the classical trajectory shape definition to cluster trajectories, but a vector with the angle between each two subsequent displacement vectors. We believe that we can group more similar trajectories by doing so and our approach can keep invariability to both scale, translation and rotation changes at the same time. More details will be given in the Section 3.

Jain *et al.* [7] improved the dense trajectory approach by decomposing visual

motion into the residual and dominant motion using an affine model. The dominant motion is assumed to be due to the camera's motion, while the residual motion is assumed to be due to the independent scene motions. With this assumption, the authors tried to compensate the camera's movement in the calculation of the optical flow and reinforce the focus in the action performed. However, the author also showed that the camera motion could not be thrown away, as it is useful as complementary information to recognize some action categories. We use a similar idea in our approach, as we also try to separate the trajectories made by the camera movement and also keep it as additional information. In his work they proposed a new descriptor called Divergence-Curl-Shear that computes local kinematic features of the optical flow along the trajectory. Unlike our approach, the final descriptor is computed using VLAD[21], an extension of the BoF.

A work presented by Wang and Schmid [8] is the current state-of-art in the field. The authors improved the dense trajectories by estimating the camera's motion and correcting it. They used SURF [22] to find feature points and match them in two consecutive frames. With that information, along the dense trajectories itself, they are capable of using RANSAC [23] to estimate the homography between those frames and assume that the transformation is made by the camera motion. To ensure that the movement done by the human(s) in the action will not interfere in the above steps, they used a human detector [24] and ignored the information extracted around those areas. After calculating the camera movement, the optical flow and, consequently, the trajectories are compensated to minimize the camera's movement influence in the extracted features. To extract the features, those areas around humans previously ignored are now taking into account and the same "visual words" extracted by Wang *et al.* [5] are now extracted around those trajectories. The final descriptor is computed using Fish Vectors [25] and the standard BoF approach.

As we can see, most of the works are focused in eliminating or compensating the camera's motion. We apply the same idea in our work, but in a simpler approach; no complex methods to detect the camera's motion are applied. Most of these recent works are also interested in using a BoF approach to form the video's descriptor. We want the descriptor of each video to be computed using exclusively its own information, so any BoF related methods are not applied either.

The presented works with the self-descriptor constraint tends to be simpler and to our best knowledge, they are all global approaches. Our work is the combination of both concepts. Although the trajectories are densely extracted, we do not use a global approach, because some of the points are not considered. Yet, we keep the self-descriptor restriction. The result of this combination is that our method can achieve much better results in comparison with other self-descriptor approaches and a fairly competitive rate against BoF methods.

3. Proposed method

In this work we use the dense trajectory method proposed by Wang *et al.* [5] to extract trajectories. Note that our approach is not bounded to the dense trajectory method and can be implemented on top of any trajectory method. As dense trajectories have shown great results in many different works [6,7,8,26], we chose it as our trajectory extractor.

3.1. *The cross product descriptor*

Each trajectory S_i^t extracted using the dense trajectory can be described as a series of points $\left\{\mathbf{p}_{0,i}^t, \mathbf{p}_{1,i}^t, \ldots, \mathbf{p}_{l,i}^t\right\}$, where i is the index of that trajectory, l is the size of the trajectory and t the frame of the first point. The trajectory's displacement vectors are then determined using the difference between two consecutive points: $\Delta\mathbf{p}_{j,i}^t = (\mathbf{p}_{j+1,i}^t - \mathbf{p}_{j,i}^t)$. We refer those displacement vectors as $\vec{v}_{j,i}$, which means that this is the j-th vector of the i-th trajectory. For simplification purposes, we are going to disregard the index t, but every step is done for every initial frame t.

To compute our descriptor for a trajectory S_i we need to find a new vector field $C_{j,i}$ of cross products. This is calculated finding, for each point $\mathbf{p}_{j,i}$ of the trajectory, the cross product between the trajectory displacement vector $\vec{v}_{j,i}$ in that point and each 3D gradient vector in a window around $\mathbf{p}_{j,i}$. As the trajectory displacement vector is composed by a vector in an optical flow field, it is expected to have two components (u, v). In order to be able to compute the cross product, we need the vector to lie in $\mathbb{R}^3$. Therefore, we add a third component to it, to represent the displacement of a trajectory in respect to the time, which is one frame, making it $(u, v, 1)$.

Thus, the vector field representing the cross product between the trajectory vector and the gradient around that point can be computed using

$$C_{j,i} = \left\{\vec{c}_{\mathbf{q}} \,|\, \mathbf{q} \in M_{\mathbf{p}_{j,i}}, \vec{c}_{\mathbf{q}} = \vec{v}_{j,i} \times \vec{g}_{\mathbf{q}}\right\},$$

where $\mathbf{q}$ is a point in the window $M_{\mathbf{p}_{j,i}}$ around the point $\mathbf{p}_{j,i}$ and $\vec{g}_{\mathbf{q}}$ is the 3D gradient in the point $\mathbf{q}$. An overview is presented in Figure 3.

The cross product between two vectors has two main properties. The first one is that the resulting vector is perpendicular to both vectors. This is an interesting property in our case because the resulting vector encodes some information about the original vectors directions at the same time. The second one is that its magnitude is related to the sine of the angle between the original vectors. If they are parallel, the magnitude is 0 and if they are perpendicular the magnitude is equal to the product of the vectors magnitudes. This is also a very interesting property in our case, since it encodes the tendency of the trajectory and the gradient to move together.

The resulting vector field $C_{j,i}$ is then quantized into a 3D histogram using a spherical coordinate system, the same way it is done in the HOG3D approach. We

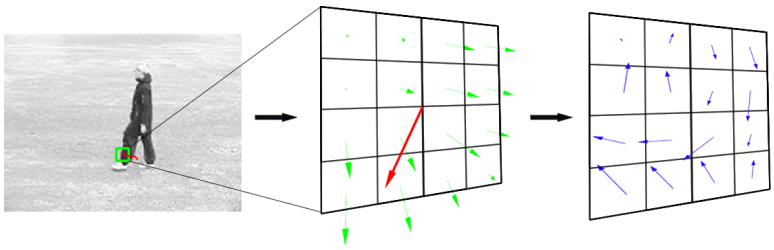

Figure 3. Overview of the cross product descriptor calculation. The picture in the left represents the frame with a trajectory tracked in a point. The picture in the middle represents the 3D gradient field (green vectors) in a surrounding window around that point and one displacement vector of the trajectory (red vector). The picture in the right is a representation of how the final vector field looks after the cross product between the trajectory displacement vector and each one of the gradients is computed.

denote $\vec{h}_{i,j}$ as a vector corresponding to the histogram of the cross product of the trajectory i in its j-th displacement vector.

The second step is to sum all histograms along the same trajectory into a single histogram $\vec{g}_i$ of all cross products of the trajectory i:

$$\vec{g}_i = \sum_{j=0}^{l-1} \vec{h}_{i,j}.$$

The resultant histogram is normalized using the L^1-norm and the final vector of information is created by concatenating the histogram $\vec{g}_i$ to the average position $(\bar{x}, \bar{y}, t)$ of all points of the trajectory in frame t:

$$\vec{f}_i = (\vec{g}_i, \bar{x}_i, \bar{y}_i, t).$$

The next step to generate the descriptor is to transform it to an orientation tensor form. This is done multiplying the vector by its transpose, given by the formula

$$\mathbf{T}_i = \vec{f}_i \vec{f}_i^{\mathsf{T}}.$$

Finally, each trajectory i has one tensor $\mathbf{T}_i$ created from the histogram of cross products and the spatio-temporal localization of that trajectory.

3.2. *Clustering trajectories by shape*

Dealing with camera motion is one of the most frequent problems in the human action recognition field. To address this problem, we propose a method to cluster trajectories based on their shape. Our hypothesis is that clustering trajectories by their shape give us the possibility to separate trajectories that are created by the

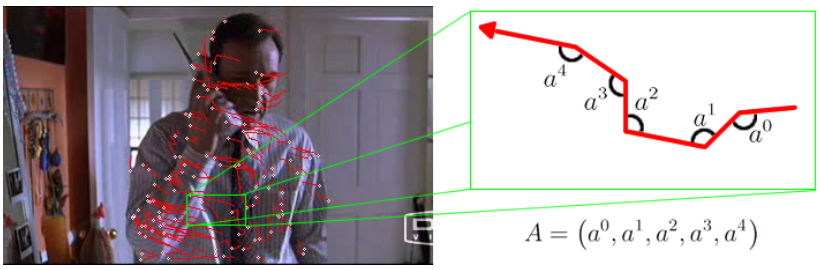

Figure 4. Illustration of how the vector of angles is computed.

camera's motion. We believe those trajectories may be characterized by having the highest angle between each connected local displacement vector and they are not discarded, as they are still source of information [7]. Additionally we also expect that, even when there is no camera movement, separating trajectories with different shapes might improve the distinguishing power of our descriptor.

In order to distinguish the trajectory by its shape, we create a vector of angles associated with each one of the trajectories. The angles are calculated between two consecutive displacement vectors of the trajectory. The vector $\vec{a}$ of the trajectory i is created using

$$\vec{a}_i = (a_{0,i}, a_{1,i}, \ldots, a_{l-2,i}),$$

where

$$a_{j,i} = \cos^{-1}\left(\frac{\vec{v}_{j,i} \cdot \vec{v}_{j+1,i}}{\|\vec{v}_{j,i}\| \cdot \|\vec{v}_{j+1,i}\|}\right). \quad (1)$$

The Equation 1 gives us the smallest angle between two connected displacement vectors in the trajectory. The resulting number is between 0 and 180 and the higher the number is, straighter is that connection. Figure 4 shows an example of how the vector is computed for a specific trajectory. After that, we use k-means to cluster the trajectories using each of its vector of angles. The mean of all angles of each cluster is then used to sort the cluster and determine how straight that group is. Classifying the clusters using this metric increase the chance that most of the camera movement will rely at the first cluster, as it tends to be straighter than trajectories generated by other movements. Also, trajectories are clustered in the same range in time. In other words, trajectories that have started at the same frame t are clustered together, discarding any information about trajectories that started over other frames. Therefore, the clustering step must be repeated for each frame of the video, which enforces the need to sort the clusters as a metric to match them between trajectories that have started at different frames.

3.3. *Calculating the final descriptor*

In order to generate the final descriptor, we used both cross product descriptor and shape-based clustering approaches. An overview of our method is illustrated in Figure 5.

First, trajectories are extracted using the dense trajectory approach. For each group of trajectories that start at a frame t, we apply the proposed strategy to cluster based on shape and sort using the mean angle of all trajectories in each cluster. The trajectories are now separated in k different groups and the tensor of the histogram of cross product $\mathbf{T}_i^t$ is calculated for each trajectory i. The next step is to sum all trajectories' tensor that belongs to the same cluster, therefore creating a new tensor for each cluster, using

$$\mathbf{R}_b^t = \sum_i \mathbf{T}_i^t \mid \boldsymbol{g}(i) = b,$$

where $\mathbf{R}_b^t$ is the tensor of the cluster $b \in [0, k-1]$ in the frame t and $\boldsymbol{g}$ is a function that determines which cluster a trajectory i belongs to, after the trajectories are already clustered by k-means. These tensors are then normalized using the L^2-norm to keep its directions discarding the magnitude. This step is required to ensure that clusters with higher population will not generate bigger tensors compared to other clusters. Next, we have k tensors for each frame in the video. Since clusters are ordered using a metric, we can match tensors from different frames. Thus, all tensors from the same cluster b from different frames are added up using

$$\mathbf{S}_b = \sum_t \mathbf{R}_b^t.$$

Finally, we have k tensors for the whole video. Each tensor $\mathbf{S}_b$ is again normalized to keep only its directions and the final descriptor $\vec{d}$ is a vector composed by the concatenation of all tensors, or

$$\vec{d} = (\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_{k-1}).$$

4. Experimental Results

To evaluate the applicability of our descriptor in the human action recognition problem, we chose two of the most used datasets in the area: KTH and Hollywood2. All clustering steps are done using a sequential k-means[a] implementation with the stopping criterion of 500 iterations or less than 0.01% of change in the cluster membership from last iteration. We used the default dense trajectory method[b] proposed by Wang *et al.* [5] to extract trajectories. Our data is classified using a

[a]Available at http://www.ece.northwestern.edu/~wkliao/Kmeans
[b]Available at http://lear.inrialpes.fr/people/wang/dense_trajectories

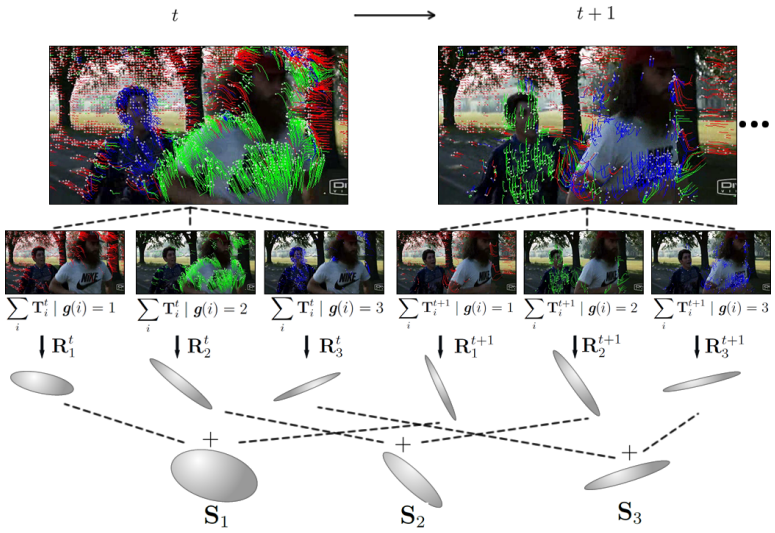

Figure 5. Overview of our method. Trajectories that start at the frame t are clustered by their shape into k clusters. This illustration shows an example for $k = 3$ clusters.

non-linear support vector machine [27], using Gaussian and Triangle kernel and a one-against-all strategy for multiclass classification.

All our tests ran under a machine with an Intel ®Xeon ®E5-4607 CPU. The machine counts with 32 GB of RAM and our method is not optimized for parallelism. For the best results, our descriptors were computed with an average of $0,343$ frames per second for the Hollywood2 dataset, and an average of $1,38$ frames per second for the KTH dataset. This time includes all steps required to generate the descriptor, including the dense trajectory extraction and both cross product and trajectory clustering strategies.

4.0.1. *KTH*

This database was proposed by Schuldt *et al.* [28] and is composed of 6 actions: walking, jogging, running, boxing, hand waving and hand clapping. These actions are performed several times by 25 people in four different scenarios: outdoors, outdoors with camera zoom, outdoors with changed clothes and indoors. All sequences are taken with homogeneous backgrounds and a static camera. The KTH dataset has a total of 2391 sequences, divided into a training set (8 people), a validation set (8 people) and a test set (9 people). The classifier is trained with the training set, the validation set is used to optimize the parameters of the classifier and the final recognition rate is based on the number of correctly predicted actions for the test set. Figure 6 shows a sample of the videos from the dataset.

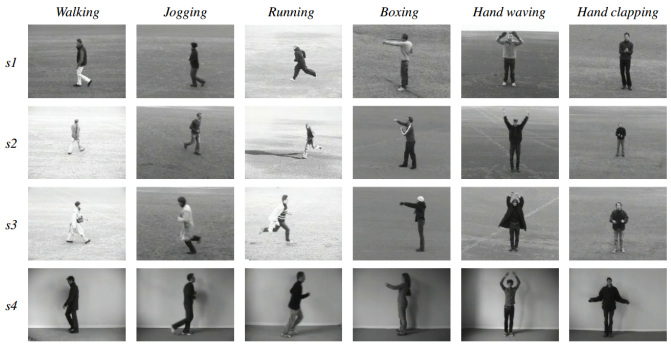

Figure 6. Illustration of KTH actions.

Figure 7. Illustration of the Hollywood2 actions.

4.0.2. *Hollywood 2*

The Hollywood2 dataset was proposed by Marszalek*et al.* [29] and is composed by 12 actions: answering phone, driving car, eating, fighting, getting out of car, hand shaking, hugging, kissing, running, sitting down, sitting up, and standing up. The sequences are made by video clips extracted from 69 different Hollywood movies and many challenges like camera movement, illumination conditions and background clutter are applied. There are a total of 1707 sequences in the dataset, which are divided in a training set with 823 samples and a test set with 884 samples. The classifier is trained with the training set and the final recognition performance is measured by the mean average precision (mAP), which is the mean of the percentage of correctly predicted actions of the test set for each class. An example of the sequences is shown in Figure 7.

4.1. *Parameters*

There are some parameters of our method that should be considered. For instance:

- Number of clusters: k represents the number of clusters the trajectories are grouped to. When $k = 1$, our approach to cluster trajectories is not applied and a single tensor is computed for all trajectories and, consequently, for the whole frame and for the entire video. There is a trade-off between increasing and decreasing the value of k. Lower values for k may lead to different movement types grouped in the same cluster. Higher values for k, not just increase linearly the size of the final descriptor but there is also a chance to cluster similar movements into different clusters;
- Histogram bins: *bin* represents the number of subdivisions in the histogram of cross product. Lower values of *bin* do not acquire the information we need and lead to a histogram that is not able to capture the cross product field tendency of direction. Higher values for *bin* does not summarize well the information and does not match well cross product fields that are similar but slightly different. Also, increasing the value of *bin* increases quadratically the size of the final descriptor so, higher numbers should be avoided;
- Window size: $M \times M$ is the size of the window around each point of the trajectory where the cross product is calculated. A small window size may not capture important features along the trajectory. On the other hand, a big window size may overlap between nearby trajectories and acquire redundant or excess of information that otherwise would be discarded.
- Sampling stride: w defines the space between points that compose the sampling grid for new trajectory points. In other words, points that are candidates to start a trajectory are sampled in a grid spaced by w pixels in both x and y dimensions. A lower number for stride increases the coverage of the method, because more points are being sampled. The downside is that more trajectories are extracted and more computation power is required to process that information. Also, sampling a lot of points does not necessarily increase the amount of meaningful information, as the trajectory windows tend to overlap. Unless otherwise stated, the value for the stride is 5, as stated in the original dense trajectory work [5];
- Cornerness of the point: the parameter q defines the minimum value for the maximum eigenvalue of a point so it can start a trajectory. A higher q increases the pressure to the candidate points, so the cornerness measure should be higher and more corner-like points are selected. In opposition, a lower q allows more points to be tracked, but increases the chance of getting points that rely in flat or edge regions and are less reliable. Our default value for q is 0.001 as stated in the original dense trajectory work [5];
- Maximum scale depth: dense trajectories are extracted in multiple scales individually to achieve some scale invariance. Increasing the maximum scale grants a better coverage of the method but also increases its computational complexity. The default value for the maximum scale depth in our methods is 5. The factor of each scale is $f = {}^{1}/_{\sqrt{2}}$, which means that each new scale has half the area of the previous scale;
- Trajectory size: l defines for how many frames the initial point will be tracked to

compose the trajectory. Lower values for l will generate short trajectories that does not encode properly the executed motion. Higher values for the trajectory size lead to longer trajectories that are more likely to drift from the original tracked point and therefore are less reliable;

- Power normalization: power normalization is a technique to equalize descriptors in order to lower high peaks on it. Peaks that are too high tend to cover information in the lower coefficients of the descriptor. To address this problem, all coefficients of the descriptor are powered by a number $\alpha \in]0, 1[$. This is the same of taking the $^1/_\alpha$-th root of each coefficient. By using this technique, coefficients that carry too much energy are reduced and the low energy coefficients are enhanced. Nevertheless, they do not lose their relationship and coefficients that are greater than others before the power normalization are still greater after it. In our experiments we used $\{0.30, 0.27, 0.25, 0.20, 0.15, 0.10\}$ for α;
- SVM kernel: we used both Triangle and Gaussian kernel on the SVM classifier in our experiments.

4.2. *Experiments*

4.2.1. *KTH Dataset*

For the KTH dataset we tested the following range of parameters: $l \in \{5, 10, 20\}$, $bin \in \{(6 \times 3), (10 \times 5), (16 \times 8), (18 \times 9), (22 \times 11), (24 \times 12)\}$, $k \in \{1, 2, 3, 5, 10\}$, $w \in \{2, 5\}$.

Table 1 shows our best result. The parameters are $k = 5$, $bin = 24 \times 12$, $l = 10$, $w = 2$, $\alpha = 0.27$ using a triangle kernel. The overall recognition rate is 94.1%. In the confusion matrix we can see the classes Boxing, Hand Clapping, Hand Waving and Walking achieved a very good recognition rate. Most of the wrong predictions are concentrated mutually between Running and Jogging and between Walking and Jogging. This is expected because in the KTH dataset, those movements are very similar and, for as much as we want the descriptor to be able to complete differentiate them, sometimes there is an intersection in the actions performed that are very hard to distinguish.

Table 1. Confusion matrix for our best result for the KTH dataset.

	Box	HClap	HWav	Jog	Run	Walk
Box	98.6%	1.4%	0%	0%	0%	0%
HClap	1.4%	98.6%	0%	0%	0%	0%
HWav	1.4%	1.4%	97.2%	0%	0%	0%
Jog	0%	1.4%	0%	87.5%	6.9%	4.2%
Run	0%	0%	0%	16.7%	83.3%	0%
Walk	0%	0%	0%	0.7%	0%	99.3%

In Figure 8 we analyze the impact of modifying each parameter in our best result. In every case, the default values for our best configuration are kept, changing

only the parameter we are evaluating. In Figure 8(a) we can see that increasing k improved the recognition rate until $k = 5$ and dropped for $k = 10$. In Figure 8(b) we vary the values for *bin*. In general, there is a tendency to improve the recognition rate as *bin* increases but, even for small values our method still has a fair performance. The trajectory size l is evaluated in Figure 8(c) and $l = 10$ showed the best result, approximately 0.5% higher recognition rate than $l = 5$ and $l = 20$. In Figure 8(d) we can see the impact of changing the stride parameter. For the best result, changing w showed a great difference. The value $w = 2$ is about 1% higher in the recognition rate than $w = 5$. We can see in Figure 8(e) that the kernel chosen to classify the data had a very significant impact in our best result. The triangle kernel showed an approximately 1.5% greater result than the Gaussian kernel.Figure 8(f) shows the impact of changing the power normalization coefficient. The value $\alpha = 0.27$ showed the best result overall but $\alpha = 0.25$ has a comparable recognition rate. At last, Figure 8(g) shows that changing the window size from its default value does not lead to great changes in the recognition rate.

4.2.2. *Hollywood2 Dataset*

For the Hollywood2 dataset we tested the following range of parameters: $l \in \{5, 10, 15\}$, $bin \in \{(6 \times 3), (10 \times 5), (16 \times 8), (22 \times 11), (24 \times 12), (26 \times 13)\}$, $k \in \{1, 2, 3, 5, 10\}$.

Table 2 shows the best recognition rate achieved by our method. The results were obtained by using $k = 5$, $bin = 26 \times 13$, $l = 5$, $w = 5$, $\alpha = 0.20$ and the triangle kernel.

Table 2. Average precision for each class of Hollywood2 for our best configuration.

Action	APhone	DCar	Eat	FPerson	GetOutCar	HShake
AP(%)	26.5	85.2	59	58	29.8	32.3
Action	HPerson	Kiss	Run	SDown	SUp	StandUp
AP(%)	24.3	48.3	66.1	54.3	19.6	52.8

In Figure 9 we can see the impact of changing the parameters for our best result. Changing the value for k in igure 9(a) shows that when $k = 1$ and our strategy to cluster trajectories is not applied the recognition rate decreases for a reasonable amount. The best result was achieved with $k = 5$ but, other values for k still reaches a good performance. The Figure 9(b) shows that increasing the number of bins in the histogram has a positive effect in the final recognition rate, although it tends to stabilize after $bin = 22 \times 11$. The trajectory size is evaluated in Figure 9(c) and $l = 5$ shows the best result over others. We can see in Figure 9(d) that the chosen kernel also had a very significant impact in our best result. The triangle kernel has an approximately 1.25% greater result than the Gaussian kernel. The Figure

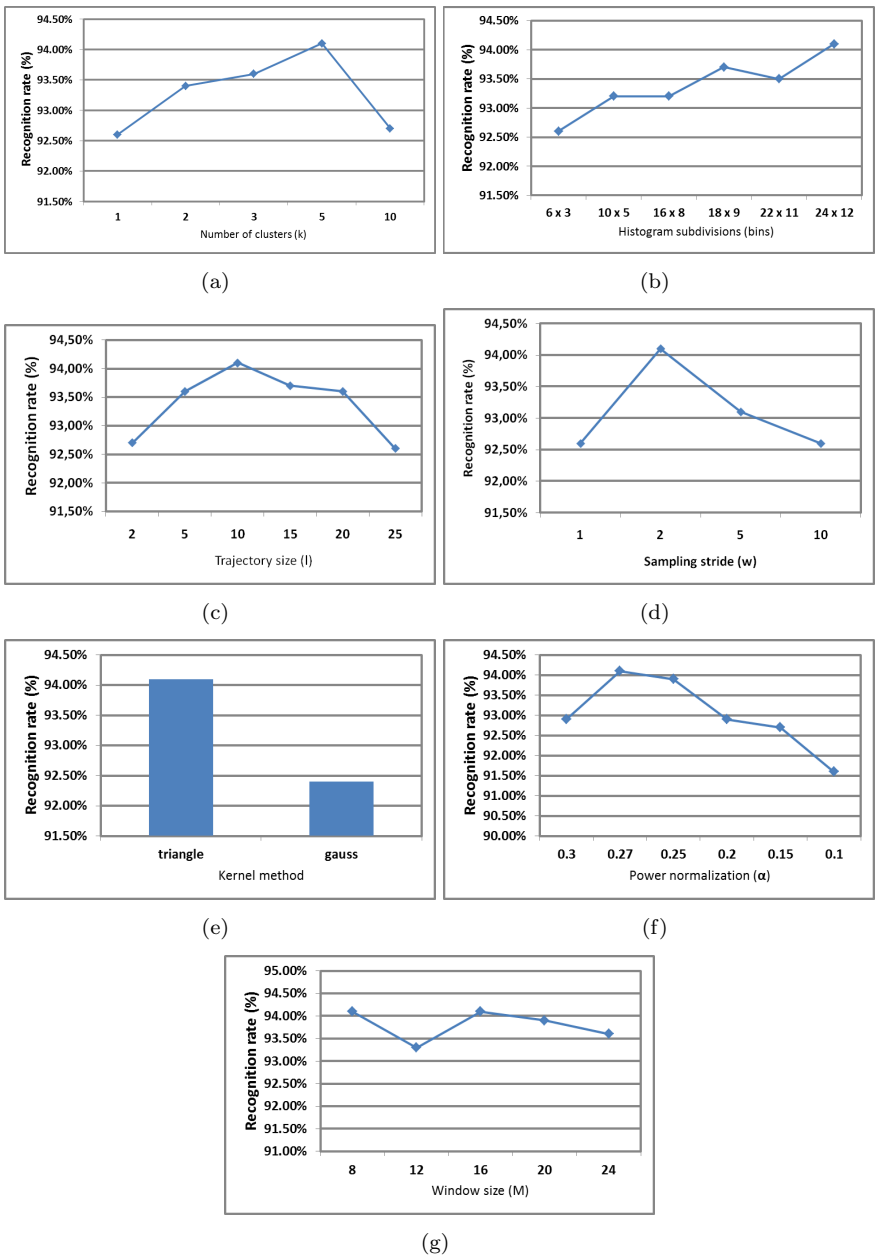

Figure 8. Impact of parameters variation in the best result for the KTH dataset.

8(e) shows the impact of changing the power normalization coefficient. The value $\alpha = 0.2$ showed the best result overall. The Figure 9(f) shows the recognition rate

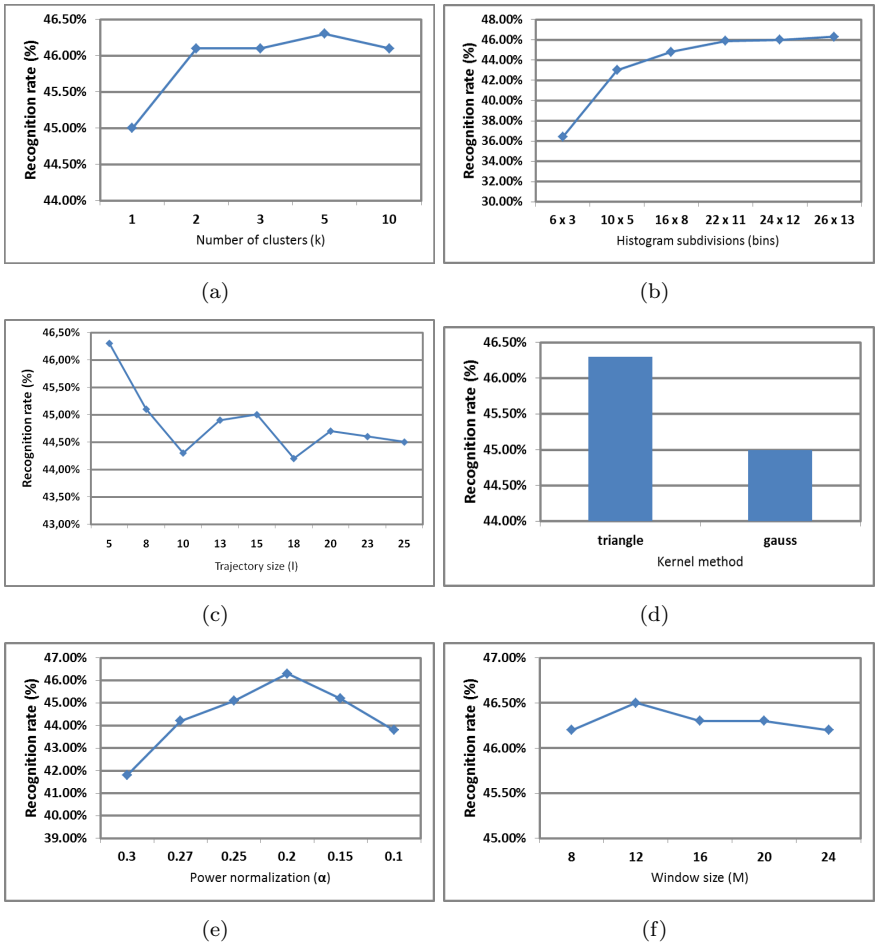

Figure 9. Impact of parameters variation in the best result for the Hollywood2 dataset.

in function of the window size. Note that using a window of 12×12 improved the best result by 0.2%, which is a significant improvement.

4.3. *Comparison with previous works*

In this subsection we compare our best results with the state-of-the-art in literature. To compare our results, we need to take into account that most of the methods are not restricted to the self-descriptor constraint and use approaches similar to the BoF. It means that their final descriptor for each video is calculated using the information of all videos merged together. The Table 3 presents a comparison between our best results and these works.

We can see that among self-descriptor works, our method achieved the best

Table 3. Comparison with state-of-the-art for KTH and Hollywood2 datasets.

	KTH	Hollywood2
Dictionary-based Methods		
Wang *et al.* [5]	94.2	58.2
Kobayashi and Otsu [30]	**95.6**	47.7
Wang *et al.* [26]	95.3	59.9
Jain *et al.* [7]		62.5
Wang and Schmid [8]		**64.3**
Self-descriptor Methods		
Perez *et al.* [16]	92.0	34.0
Mota *et al.* [17]	92.5	40.3
Sad *et al.* [31]	93.3	41.9
Mota *et al.* [18]	93.2	40.3
Figueiredo *et al.* [12]	87.7	34.9
Our method	**94.1**	**46.3**

performance for both KTH and Hollywood2 datasets. For the KTH, we achieve recognition 0.8% better than the previous best result for self-descriptor methods and a comparable result against dictionary-based methods. For the Hollywood2, we improved the best result for self-descriptor methods by 4.4%. We believe this great improvement comes from the fact that the Hollywood2 is a challenging dataset as it contains several camera movement and many different scenarios. In this particular situation, our method to cluster trajectories can separate those movements quite well, which is not done in the previous self-descriptor works. The same impact cannot be seen in the KTH dataset because it does not contain much camera movement and the background is very homogeneous. Although we had a great improvement for the Hollywood2, it is possible to see that our method is still not able to reach the current state-of-start methods that are based on a dictionary of visual words.

5. Conclusion

In this work, we proposed a method to calculate a descriptor for videos, based on the histogram of the cross products between trajectories and 3D gradients. We also propose a strategy to cluster trajectories based on their shape. Both contributions have improved the recognition rate in two known datasets, achieving, to our best knowledge, the highest recognition rate among self-descriptor methods and a fairly recognition rate compared with dictionary-based methods.

Our results showed that for the Hollywood dataset, the best setup can achieve 46.3% in the mean average precision, using 10 trajectory's clusters, trajectories with 5 frames long and 26×13 subdivisions in the histogram of cross product. For the KTH dataset, the best setup can achieve 94.1% using 5 trajectory's clusters,

trajectories with 10 frames long, 24×12 subdivisions in the histogram of cross product and sampling trajectories in a 2 pixel spaced grid.

One suggestion for future works is to cluster all trajectories from the whole video just one time. This is different from the strategy to cluster trajectories applied in this work, where we cluster trajectories that start at the same frame. This suggestion might be a good idea because our strategy to match clusters from different frames using the mean of its angles may not always work. For example, consecutive frames with very different motion patterns will end up to be summed in the same tensor.

In our tests we noticed that very low values for the power normalization showed the best results. This might be an effect of dealing with tensors in the Euclidean space, instead of the Log-Euclidean space. Thus, further studies using Log-Euclidean, as done by Yuan *et al.* [14] are recommended.

Acknowledgements

Thanks to CAPES and UFJF for funding.

Bibliography

1. P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th International Conference on Multimedia MULTIMEDIA '07*, pp. 357–360, (ACM, New York, NY, USA 2007).
2. A. Klaser, M. Marszalek and C. Schmid, "A Spatio-Temporal Descriptor Based on 3D-Gradients," in *BMVC 2008 - 19th British Machine Vision Conference*, pp. 275:1–10, (British Machine Vision Association, Leeds, United Kingdom, 2008).
3. G. Willems, T. Tuytelaars and L. Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," in *Proceedings of the 10th European Conference on Computer Vision: Part II, ECCV '08*, pp. 650–663, (Springer-Verlag, Berlin, Heidelberg 2008)
4. I. Laptev, "On space-time interest points," *Int. J. Comput. Vision*, **64**(2-3), 107–123 (2005).
5. H. Wang, A. Klaser, C. Schmid and C.L. Liu, "Action recognition by dense trajectories," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 3169–3176 (2011).
6. Y. G. Jiang, Q. Dai, X. Xue, W. Liu and C. W. Ngo, "Trajectory-based modeling of in *Proceedings of the 12th European Conference on Computer Vision - Volume Part V, ECCV'12*, pp. 425–438, (Springer-Verlag, Berlin, Heidelberg, 2012).
7. M. Jain, H. Jégou and P. Bouthemy, "Better exploiting motion for better action recognition," in *CVPR - International Conference on Computer Vision and Pattern Recognition*, (Portland, États-Unis, 2013).
8. H. Wang, and C. Schmid, "Action Recognition with Improved Trajectories," in *ICCV 2013 - IEEE International Conference on Computer Vision*, pp. 3551–3558, (IEEE, Sydney, Australia, 2013).
9. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1*, pp. 886–893 vol. 1 (2005).
10. I. Laptev, M. Marszalek, C. Schmid and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8 (2008).

11. N. Dalal, B. Triggs and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Proceedings of the 9th European Conference on Computer Vision - Volume Part II, ECCV'06*, pp. 428–441, (Springer-Verlag, Berlin, Heidelberg,2006)
12. A.M. Figueiredo, H.A. Maia, F.L. Oliveira, V.F. Mota and M.B. Vieira, "A video tensor self-descriptor based on block matching," in *Computational Science and Its Applications ICCSA 2014*, *Lecture Notes in Computer Science*, vol. 8584, pp. 401–414 (Springer International Publishing, 2014).
13. G.L. Kindlmann, "Superquadric tensor glyphs," in *Proceedings of the Sixth Joint Eurographics - IEEE TCVG Conference on Visualization*, pp. 147–154, (Eurographics Association, 2004).
14. C. Yuan, W. Hu, X. Li, S. Maybank and G. Luo, "Human action recognition under log-euclidean riemannian metric," in *Proceedings of the 9th Asian Conference on Computer Vision - Volume Part I, ACCV'09*, pp. 343–353, (Springer-Verlag, Berlin, Heidelberg, 2010).
15. P. Dollar, V. Rabaud, G. Cottrell and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*, pp. 65–72 (2005).
16. E.A. Perez, V.F. Mota, L.M. Maciel, D.O. Sad and M.B. Vieira, "Combining gradient histograms using orientation tensors for human action recognition," in *ICPR*, pp. 3460–3463. (IEEE, 2012).
17. V. Mota, J. Souza, A. De Araujo and M.B. Vieira, "Combining orientation tensors for human action recognition," in *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI - Conference on*, pp. 328–333 (2013).
18. V.F. Mota, E.A. Perez, L.M. Maciel, M.B. Vieira and P.H. Gosselin, "A tensor motion descriptor based on histograms of gradients and optical flow," *Pattern Recogn. Lett.* **39**, 85–91 (2014).
19. V.F. Mota, E.A. Perez, M.B. Vieira, L.M. Maciel, F. Precioso and P.H. Gosselin, "A tensor based on optical flow for global description of motion in videos," in *Proceedings of the 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images, SIBGRAPI '12*, pp. 298–301, (IEEE Computer Society, Washington, DC, USA, 2012).
20. D. Picard and P.H. Gosselin, "Improving Image Similarity With Vectors of Locally Aggregated Tensors," in *2011 IEEE International Conference on Image Processing (IEEE ICIP2011)*, pp. 669 – 672. (Brussels, Belgium, 2011).
21. H. Jegou, M. Douze, C. Schmid and P. Perez, "Aggregating local descriptors into a compact image representation," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 3304–3311 (2010).
22. H. Bay, A. Ess, T. Tuytelaars , L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.* **110**(3), 346–359 (2008).
23. M.A. Fischler and R.C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM* **24**(6), 381–395 (1981).
24. A. Prest, C. Schmid and V. Ferrari, "Weakly supervised learning of interactions between humans and objects," *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(3), 601–614 (2012).
25. F. Perronnin, J. Sánchez and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proceedings of the 11th European Conference on Computer Vision: Part IV, ECCV'10*, pp. 143–156. (Springer-Verlag, Berlin, Heidelberg, 2010).
26. H. Wang, A. Kläser, C. Schmid and C.L. Liu, "Dense trajectories and motion bound-

ary descriptors for action recognition," *Research Report RR-8050*, (INRIA, 2012).
27. C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.* **20**(3), 273–297 (1995).
28. C. Schuldt, I. Laptev and B. Caputo, "Recognizing human actions: A local svm approach," *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3* (ICPR'04), (2004), p. 32–36.
29. M. Marszalek, I. Laptev and C. Schmid, "Actions in context,", *Computer Vision and Pattern Recognition* (CVPR), (2009), p. 2929–2936.
30. T. Kobayashi and N. Otsu, "Motion recognition using local auto-correlation of space-time gradients," *Pattern Recogn. Lett.* **33**(9), 1188–1195 (2012).
31. D. Sad, V.F. Mota, L.M. Maciel, M.B. Vieira and A.D.A. Araújo, "A tensor motion descriptor based on multiple gradient estimators," in *Proceedings of the 2013 XXVI Conference on Graphics, Patterns and Images, SIBGRAPI '13*, pp. 70–74. (IEEE Computer Society, Washington, DC, USA, 2013).

Photo and Bibliography

Felipe Caetano obtained a graduate degree in Computer Science from Universidade Federal de Juiz de Fora (2012) and M.Sc. in Computer Science in the same university (2014). He is currently a Information Technology analyst at the Organizational Knowledge Management Center of the Universidade Federal de Juiz de Fora.

Marcelo Vieira He obtained a graduate degree in Computer Science from Pontifícia Universidade Católica de Minas Gerais (1995), M.Sc. in Computer Science from Universidade Federal de Minas Gerais (1998), Ph.D. in Image & Signal Processing from ENSEA/Université de Cergy-Pontoise in France (2002) and Ph.D. in Computer Science from Universidade Federal de Minas Gerais (2002), performed in conjunction. He is currently associate professor at Universidade Federal de Juiz de Fora. His main knowledge area is Computer Science, working on the following subjects: fluid dynamics, tensor fields and scientific visualization, physical systems geometric modeling, 3d reconstruction and pattern recognition. He is coordinator of the Group for Computer Graphics, Image and Vision at UFJF.

Rodrigo Silva obtained a graduate degree in Computer Science from Universidade Católica de Petrópolis (1999), M.Sc. in Systems Engineering and Computer Science from Universidade Federal do Rio de Janeiro (2002), Ph.D. in Civil Engineering from Universidade Federal do Rio de Janeiro (2006) and Postdoctorate in Computer Science from Laboratório Nacional de Computação Científica (2008). He is currently adjunct professor at Universidade Federal de Juiz de Fora and his main knowledge are is Computer Graphics with emphasis in Augmented Reality and Virtual Reality.