A Variable Size Block Matching Based Descriptor for Human Action Recognition

Fábio L. M. de Oliveira Helena A. Maia Virgínia F. Mota Marcelo B. Vieira and Arnaldo de A. Araújo

***Abstract*—In this work we present a research which integrates the Block Matching technique into the Human Action Recognition field. We propose a method to create a video descriptor composed of three main steps: Computation of displacement vectors between frames with a Variable Size Block Matching Algorithm, histogram representation of these vectors, and orientation tensor generation from the histogram. It is a promising approach for human action recognition since it is a simple method with low computational cost which achieves comparable results with state-of-the-art techniques.**

***Index Terms*—Human Action Recognition, Block Matching, Self-descriptor, Tensor Descriptor.**

I. Introduction

DETECTING movement in a sequence of images is an important research field of computer vision. Several applications, such as surveillance, video indexing and human action recognition, rely on the quality and efficiency of methods carrying out this task. Block Matching is one of these methods and consists in tracking the movement of rectangular regions between frames of a video. Although largely used for video compression, Block Matching has yet to be explored in the Human Action Recognition context as a low computational cost alternative to optical flow or gradients.

The assumption is that if there were continuous motion in an image sequence, several blocks from one image could be found on the next one, but in different positions. This was described as a "piecewise translation" by Jain and Jain [1], who first introduced the technique. Thus, the goal would be to find where these blocks, possibly representing objects, are in the following frame.

Block Matching Algorithms (BMA) vary mainly on search strategies and error functions. Search strategies are more efficient ways to analyze an image in order to find the best match for a region. Since it is highly unlikely that two regions will have exactly the same pixel intensities, an error function is used as criterion for which block provides the best match, measuring how similar these regions are.

There are several BMA variants which can be used to extract motion information. One such variant is the Variable Size Block Matching Algorithm (VSBMA), which does not maintain a fixed size for the blocks analyzed throughout the computation. VSBMA can employ the same search strategies and error functions as BMA, but the sizes of the blocks often change via split or merge, depending on results yielded by the block matching core routine.

In this work, we propose a video descriptor using VSBMA. The Block Matching method runs fast and can potentially generate compact descriptors since it is widely used in video compression. Moreover, it is a simple method compared to other approaches to extract motion information like 3-D gradients, because it yields a more coarse representation, with less vectors per frame. As we use variable block sizes, we are able to cover more homogeneous regions and to avoid redundancies. We show that the computational cost does not increase prohibitively versus the fixed block size approach.

The following subsections are dedicated to related works and technique overview. Section II presents the technical background with a brief explanation of VSBMA. In Sec. III, we detail each step of our method. We show the experiments conditions in Sec. IV and the results and discussions in Sec. V.

A. Related Work

Human action recognition has been an active field of research over many years now. It consists of classifying motion in videos and can be divided into two tasks: motion extraction and motion representation.

Several distinct techniques have been employed to extract motion, like optical flow [2] and 3D gradients [3]. But so far, block matching has not been thoroughly explored as an action recognition tool, even though it has been used in an array of applications where motion is a relevant feature. Amel et al. [4] use motion estimation to detect shot boundaries in video sequences. Hafiane et al. [5] present a method for video registration based on block matching. Over the years, there has been several works employing VSBMA, especially for video encoding and compression [6], [7], [8].

To represent the motion extracted, the most common structure used is the histogram [3], [9], [10], [11]. Histograms are interesting for video description as they are simple structures which carry a compact representation of the motion information. In Mota et al. [10], the final descriptor is an orientation tensor generated from a Histogram of Oriented Gradients (HOG). Tensors are robust mathematical tools and good aggregators. They can capture the local orientation and

The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Vanessa Testoni.

Fábio L. M. de Oliveira, Helena A. Maia and Marcelo B. Vieira are with Departamento de Ciência da Computação, Universidade Federal de Juiz de Fora, Juiz de Fora - MG, Brasil. Emails: {fabio, helena.maia, marcelo.bernardes}@ice.ufjf.br.

Virgínia F. Mota and Arnaldo de A. Araújo are with Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte - MG, Brasil. Emails: {virginiaferm, arnaldo}@dcc.ufmg.br.

This paper is part of the Special Issue on Vision-based Human Activity Recognition.

Digital Object Identifier 10.14209/jcis.2015.11

uncertainties of motion. Thus, they could carry more useful information than a histogram.

In this work, an orientation tensor-based descriptor for videos is generated from the output of a Variable Size Block Matching Algorithm. The vector map generated from the block matching routine is accumulated on a histogram of directions and then coded into orientation tensors.

B. Technique Overview

The motion descriptors are obtained through three steps depicted in Fig. 1. The first one is to calculate the displacement vectors with the block matching algorithm. The second step is to convert these vectors into polar coordinates and build a histogram $\vec{h}$, where each bin represents an angle interval. The third step is to calculate an orientation tensor from $\vec{h}$, to serve as a condensed representation of the motion between a pair of frames. The tensors for each pair of frames and for each video sequence in the dataset are then accumulated and normalized with L_2 Frobenius norm, so that it is possible to compare different video sequences regardless of their length or resolution.

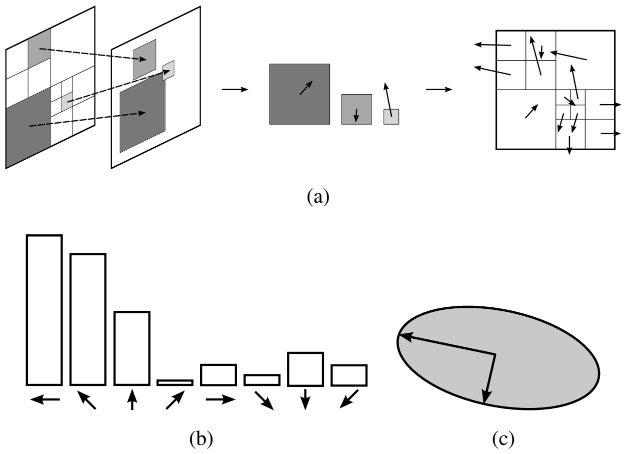


Fig. 1. Technique overview. (a) From left to right: Example of 3 blocks and their matches in the following frame yielded by the variable size block matcher. Displacement vectors obtained from said match. Vector map generated after matching all blocks in a frame. (b) Vectors accumulated into a histogram of directions. (c) Orientation tensor built based on such histogram. The ellipse is merely an illustration since generally tensor dimension is greater than 2.

II. Technical Background

This section presents the Variable Size Block Matching Algorithm [6], [7], and a brief explanation of each of its steps. During the block matching process, one frame is called "reference frame" and the following is called "target frame". In the following algorithm, they are identified as f and g, respectively.

```
1: **Input:** Frames f,g
2: **Output:** Vector set D
3: Divide frame $f$ into a set of blocks $B$
4: **for all** $b^f \in B$ **do**
5:     $b^* \leftarrow b^f$
6:     $cost_{b^*} \leftarrow \infty$
7:     $C \leftarrow$ SearchStrategy$(g, W, C, b^*)$
8:     **while** $C \neq \{\}$ **do**
9:         **for all** $b \in C$ **do**
10:            $cost_b \leftarrow \epsilon(b^f, b)$
11:            **if** $cost_b < cost_{b^*}$ **then**
12:                $b^* \leftarrow b$
13:                $cost_{b^*} \leftarrow cost_b$
14:                $(d_x, d_y) \leftarrow (b_x - b^f_x, b_y - b^f_y)$
15:            **end if**
16:        **end for**
17:        **if** $cost_{b^*} > threshold$ **then**
18:            $B, C \leftarrow$ Split$(b)$
19:        **end if**
20:        $C \leftarrow$ SearchStrategy$(g, W, C, b^*)$
21:    **end while**
22:    $D \leftarrow D \cup \{(d_x, d_y)\}$
23: **end for**
```

The algorithm consists of a series of minimization processes of a function ϵ, one for each block b^f in the reference frame f. Each block is a contiguous square set of pixels, or sub-image and the upper-leftmost pixel of a block is used as its position reference. The search space for each one of these minimization problems is bounded by the search window W and is explored through a search strategy.

In the algorithm presented, *SearchStrategy* is assumed to be able to create and/or update a set of candidate blocks C from the image g, for each iteration of the search. This update is based on the search window, on the current candidate set, and on the current best matching block. The cost function ϵ is evaluated for every candidate b in C, and the displacement vector $d(d_x, d_y)$ is the difference between the reference frame block $b^f(b^f_x, b^f_y)$ and the block $b^*(b^*_x, b^*_y)$ which minimizes function ϵ.

The function *Split* divides the block b passed as parameter into four smaller blocks and updates both the reference blocks and the candidates lists. These smaller blocks can be further divided into even smaller blocks, until they fall below a fixed error threshold or a minimum block size is reached. This way, a quad-tree structure emerges, with leaf nodes corresponding to blocks of varying sizes [7]. This tree is used in order to properly code the segmentation of the image, as shown on Fig. 2. The goal is to make the edge of the blocks coincide with the borders of the objects in the scene, forming regions with uniform intensity, just like in picture segmentation [12].

When the candidates set C is empty, meaning that the search has finished, the algorithm proceeds to the next block in B, until all blocks from the reference frame f have been matched with a block from the target frame g. VSBMA's output is the displacement vector set D, which contains a translation vector d for each block $b^f \in B$.

Fig. 3 shows a frame with displacement vectors computed through VSBMA drawn over it. Hotter colored vectors correspond to bigger blocks and colder colored vectors correspond to smaller blocks. The size of the vectors are proportional to their norms. This example suggests that the motion of more homogeneous regions of the image can be represented by a single vector, while more detailed regions need more vectors in order to properly represent its motion.

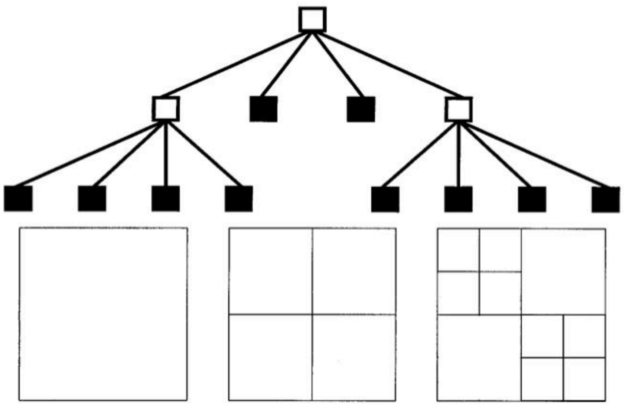


Fig. 2. Quad tree image segmentation (adapted from [13]) yielded by VSBMA. Leaf nodes correspond to blocks of various sizes.

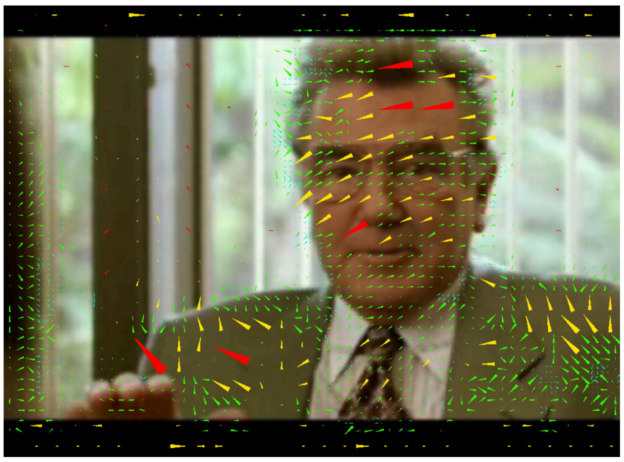


Fig. 3. VSBMA displacement vectors. Hotter colored vectors correspond to bigger blocks and colder colored vectors correspond to smaller blocks. The size of the vectors are proportional to their norms.

A. *Search Window*

The search window is merely a range of pixels in both directions, centered around the block from the reference frame which is being matched, which limits the search to a close neighbourhood of said block. Only blocks positioned within the search window are to be candidates for matching. This restriction is imposed in order to reduce computational cost of analyzing a whole frame.

It is important to note that the search window limits the size of the motion vectors calculated through the algorithm, since all the candidates for matching are assumed to be somewhat close to reference frame block. Due to this limitation, the algorithm is incapable of capturing abrupt motion in the small frame time interval.

In our implementation, we established a 15×15 pixel search window. This means that the largest vector that could result from a match would be $(\pm 7, \pm 7)$.

B. *Search Strategies*

The search strategy is the main part of the Block Matching Algorithm. It is the series of steps, or an algorithm *per se*, through which the search window is explored in order to find the best match for a block.

The main goal of a search strategy is to reduce the computational effort of the BMA retaining the quality of the results compared to examining each block in the search window.

Many strategies have been proposed over the years, such as New Three Step Search [14], Four Step Search (4SS) [15], Simple Efficient Search (SES) [16], Diamond Search (DS) [17], and Adaptive Rood Pattern Search (ARPS) [18]. In this work, we employ 4SS, which is a fast steepest descent method that yields results comparable to the exhaustive approach. The Exhaustive Search, which is the brute force approach, and 4SS are briefly explained in the next paragraph.

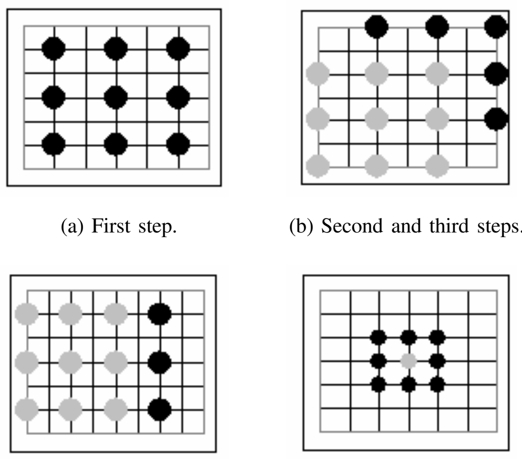
Exhaustive Search: The Exhaustive Search (ES) is the simplest and most thorough strategy. Once a search window is established, the full search consists of trying to match every possible block inside this window. For example, in a 15×15 window, 225 block comparisons are necessary in order to find the best match. ES is the most expensive strategy, but it guarantees that the best match found indeed minimizes the error function in the search window. Even so, this guarantee can have a bias in certain cases. When two candidate blocks present the same error values, the search is biased by the order in which the blocks are evaluated, retaining the upper-leftmost or the last evaluated, for instance.

Four Step Search: The Four Step Search (4SS) [15] is a steepest descent based strategy. It consists of four different search patterns used during its four steps. Starting from the center of a 15×15 window, the first step looks at 9 locations in a 5×5 window. At any step, if the minimum error is found at the center of search pattern the search jumps to fourth step. If the minimum error is at one of the eight locations except the center, then this location becomes the search origin and the search moves to the second step. The search pattern is still maintained as 5×5 pixels wide. Depending the minimum error location, 4SS might end up checking errors at 3 or 5 additional locations. If the minimum error is found at a corner, the second step checks its 5 neighbours that have not been checked on the first step. If the minimum error is at the side, the second step checks its 3 neighbours. The third step is exactly the same as the second step, except that it always leads to the fourth step. In the fourth step the pattern size is shrunk to 3×3. The location with the minimum error is the best matching block and the motion vector is set to point to that position. Especially when compared to ES, 4SS represents a big leap in terms of efficiency. In the best case scenario, only 17 block comparisons are required and in the worst case scenario, 27 block comparisons are required out of the 225 comparisons required to fully examine the 15×15 window. The patterns are shown in Fig. 4. Fig. 5 shows an example procedure.

C. *Error Functions*

The error function is the measure of similarity between two blocks. This function is defined in terms of the pixel intensities in the pair of blocks being analyzed.

A variety of error functions can be found in the literature being applied as error criteria for Block Matching Algo-



(a) First step. (b) Second and third steps.

(c) Second and third steps. (d) Fourth step.

Fig. 4. 4SS patterns [15]. Figure from [19]. Black dots represent required block comparisons on each step. Grey dots represent block comparisons already done during an earlier step.

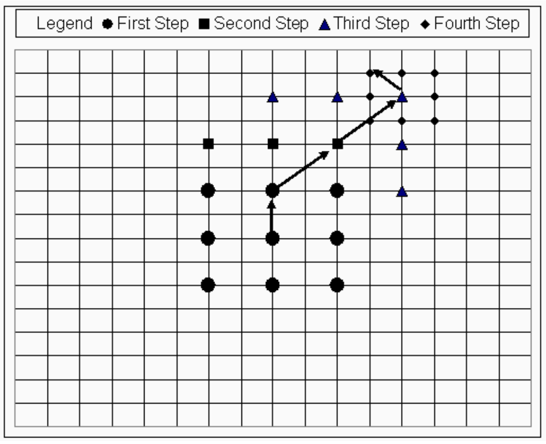

Legend ● First Step ■ Second Step ▲ Third Step ◆ Fourth Step

Fig. 5. Example routine of 4SS steps [15]. Figure from [19]. Each marker represents block comparisons during each step. Black dots for the first step, black squares for the second, blue triangles for the third, black diamonds for the last step. Vectors point to minimum error positions on each step. Final displacement vector is the sum of all shown vectors.

rithms [1], [7], [20], differing in complexity, outlier handling, topology, among other features. In this work, the Sum of Absolute Differences (SAD) and Mean Absolute Differences (MAD) were the error functions of choice for earlier experiments. Since the difference between these two functions was little, in terms of efficiency and quality, further tests and parameter tuning were made considering just SAD as error function.

In both BMA and VSBMA, the error function ϵ is the sole criterion used to find the best match for a block. The best match is the one that minimizes ϵ within the search window.

III. Generating the Descriptor

A. Block Matching Phase

As shown in Sec. II, each video frame f is subdivided into non-overlapping blocks with an initial block size. When the video resolution is not a multiple of the block size, blocks may encompass regions out of the bounds of a frame. In this case, the intensity value for an out-of-bounds pixel is the same as its closest neighbour in the borders of the image, creating a stretching effect. Fig. 6 illustrates this scenario.

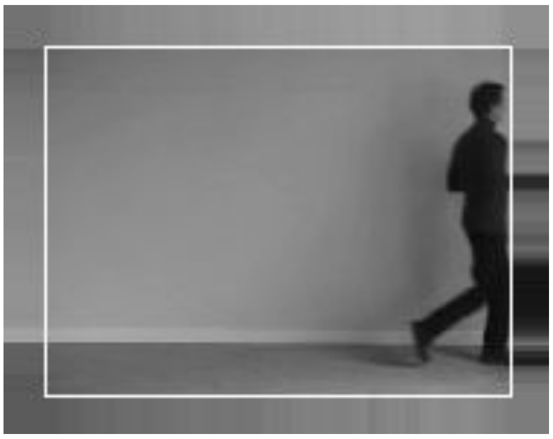


Fig. 6. Treatment for out-of-bounds block coordinates. The white rectangle highlights the original frame. Intensity values for pixels outside of this rectangle are the same as those in the border.

The 4SS method then finds the block which minimizes the error function. If the error is greater than a threshold, the block is split until the error is below the threshold, or the block reaches the minimum size of 4×4 pixels. The result of this method is a displacement vector $d(i,j) = (d_x, d_y)$ for each block, where (i,j) are the block indexes. These vectors are converted to equivalent polar coordinates $c(i,j) = (\theta, r)$ with $\theta = \tan^{-1}(\frac{d_y}{d_x})$, $\theta \in [0, 2\pi]$ and $r = \| d(i,j) \|$.

B. Histogram of Directions

A motion estimation histogram is used as a compact representation of the motion vector field obtained from each frame. It is defined as the column vector $\vec{h}_f(h_1, h_2, \ldots, h_{n_\theta})^T$, where n_θ is the number of cells for the θ coordinate. We use a uniform subdivision of the angle intervals. Each interval is populated as the following equation:

$$h_l = \sum_{i,j} r(i,j) \cdot \omega(i,j) \ , \tag{1}$$

where $l = 1, 2, \ldots, n_\theta$ and $\omega(i,j)$ is a vector weighting factor, which is a Gaussian function with $\sigma = 0.01$ in our experiments. The Gaussian function is used to attenuate the effect of having strict bin boundaries. When a vector has an orientation which is close to one of these boundaries, its magnitude is spread amongst neighboring bins. This way, we take into account any uncertainty regarding to which bin said vector should be assigned. The whole frame vector field is thus represented by a vector $\vec{h}_f$ with n_θ elements.

C. Tensor Descriptor

An orientation tensor is a representation of local orientation which takes the form of a $n \times n$ real symmetric matrix for n-dimensional signals [21]. Given a vector $\vec{v} \in \mathbb{R}^n$, it can be represented by the tensor $\mathbf{T} = \vec{v}\vec{v}^{\mathrm{T}}$. Then, we use the orientation tensor to represent the histogram $\vec{h}_f \in \mathbb{R}^{n_\theta}$. The frame tensor, $\mathbf{T}_f \in \mathbb{R}^{n_\theta \times n_\theta}$, is given by:

$$\mathbf{T}_f = \vec{h}_f \cdot \vec{h}_f^{\mathrm{T}} \ . \tag{2}$$

Individually, these frame tensors have the same information as $\vec{h}_f$, but several tensors can be combined to find component covariances.

D. Orientation Tensor

The motion average of consecutive frames can be expressed using a series of tensors. The average motion is given by

$$\mathbf{T} = \sum_{f=1}^{n_f} \frac{\mathbf{T}_f}{\| \mathbf{T}_f \|_2} \ ,$$

using all n_f video frames.

By normalizing $\mathbf{T}$ with a L_2 norm, we are able to compare different video clips or snapshots regardless their length or image resolution. Since $\mathbf{T}$ is a symmetric matrix, it can be stored with $d = \frac{n_\theta(n_\theta+1)}{2}$ elements.

If the motion captured in the histograms are too different from each other, we obtain an isotropic tensor which does not hold useful motion information. But, if accumulation results in an anisotropic tensor, it carries meaningful average motion information of the frame sequence [10].

IV. Experiments

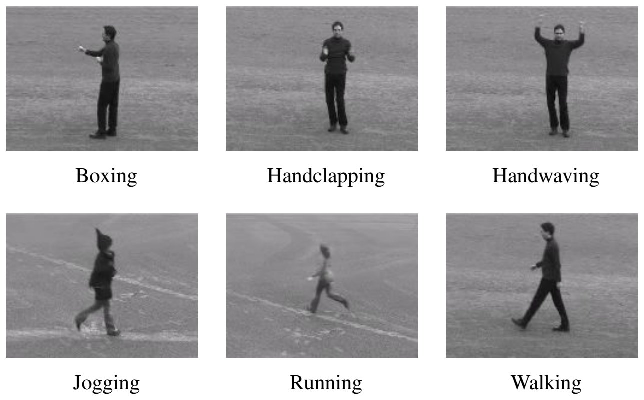


Fig. 7. Example videos from KTH dataset action categories [22]. Second row shows some of the different actors and camera viewpoints found in the database.

The experiments were made on KTH dataset [22](Fig. 7), which contains 600 videos of six human actions: walking, running, jogging, boxing, hand waving and hand clapping. These actions are performed by 25 people in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes, and indoors. As in [22], the dataset is split in about four sequences for each video, producing a total of $2,391$ files. The sequences have a resolution of 160×120 pixels and 25fps frame rate.

This dataset provides video samples with only one action portrayed. No object or person other than the actor performing the intended action appears in the same sequence. This property is appropriate for our global descriptor, since all the movement in the scene is taken into account.

We use a SVM classifier to evaluate our descriptor on KTH. All tests were run on an Intel®Core$^{\mathrm{TM}}$2 Quad Q9550 2.83GHz with 4GB memory running a single thread per video.

Recognition Rates: The quality measure used is the output of a SVM classifier, which takes the descriptors for the whole database and divides them into two fixed groups: a training set and a test set. Aside from this division, we follow the same classification protocol as [22]. The classifier produces 6 recognition rates for each block matching parameter combination, 3 using a triangular kernel, and 3 using a Gaussian kernel. From these results, we take the highest ones achieved and present them in Sec. V.

Efficiency: In order to make an efficiency assessment, two criteria are considered: frame rate, and number of blocks per frame, per video. The frame rate serves as running speed measurement, while the number of blocks measures memory efficiency. The number of blocks can also be related to running speed, since the split operations required to create more blocks also require function calls and block comparisons.

V. Results and Discussion

A. Recognition Rates

The accuracy shown in Tables I and III refer to the highest percentage of correct action predictions obtained from the descriptor generated from BMA and VSBMA, respectively.

TABLE I
EXPERIMENTS WITH DIFFERENT BLOCK SIZES USING METHOD BASED ON BMA. ACCURACIES REFER TO THE HIGHEST PERCENTAGE OF RIGHT PREDICTIONS ACHIEVED BY THE SVM CLASSIFIER. BEST RESULTS ARE SHOWN IN BOLD.

Block Size	Strategy	Accuracy
8	Exhaustive	73.9
8	**4SS**	**79.3**
16	**Exhaustive**	**79.2**
16	4SS	78.1
24	Exhaustive	76.8
24	**4SS**	**79.6**
32	Exhaustive	75.1
32	4SS	75.1

Table I shows the recognition rates for all BMA test runs. Best results are shown in bold. For the benefit of comparison, we also include exhaustive search accuracy results in this first experiment. No other experiments were run using ES, as its low speed makes thorough parameter exploration not feasible. These recognition rates indicate that block size plays a major role on the descriptor's accuracy, since there is only one block size for all blocks in regular BMA. The case with block size 32 achieves poor results due to the big blocks encompassing too heterogeneous regions, and thus failing to capture the detailed motion within. The case with block size 8 shows two very different results, indicating that even though the block size may be appropriate to capture fine motion, it may also be more sensitive to noise and compression artifacts, and thus misleadingly capturing background motion. The cases with block sizes 16 and 24 seem to strike a balance between the two previous cases, achieving the best results for BMA.

For VSBMA tests, one more parameter is considered, the error threshold. Also, since it is the focus of this work, we have conducted experiments with 8 block sizes and 8 threshold values, to a total of $8 \times 8 = 64$ different parameter settings. Table II shows the parameter values used on our experiments with VSBMA.

TABLE II
PARAMETER VALUES FOR EXPERIMENTS WITH VSBMA.

Parameter	Values
Block Size	8, 16, 24, 32, 40, 48, 56, 64
Threshold	1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000

The values for block sizes were chosen based on the dataset resolution of 160×120 pixels, H.264 (HEVC) specification of block sizes ranging from 8×8 up to 64×64 [23], and recognition rates from preliminary experiments that showed a decrease in quality for larger blocks. As for threshold values, the choice was based on the maximum error value ($block\_width \times block\_height \times \#channels \times 255$), and on the apparently decreasing power law distribution of error values suggested also by preliminary experiments.

Table III shows a summary of the recognition rates obtained with VSBMA. Once again, best results are shown in bold. Compared to its BMA counterparts, VSBMA tests show increments on recognition rates ranging from 1% up to 15%. This is a solid improvement, considering VSBMA still retains BMA's real time computation capability.

TABLE III
VSBMA EXPERIMENTS WITH VARIOUS INITIAL BLOCK SIZES AND THRESHOLD VALUES. ACCURACIES REFER TO THE HIGHEST PERCENTAGE OF RIGHT PREDICTIONS ACHIEVED BY THE SVM CLASSIFIER. BEST RESULTS ARE SHOWN IN BOLD.

Block Size	Threshold	Accuracy
8	1000	80.4
8	2000	80.7
16	**2000**	**86.0**
16	4000	85.1
24	2000	85.2
24	**4000**	**86.0**
32	**2000**	**86.7**
32	4000	85.8
40	**4000**	**86.2**
40	**8000**	**86.6**
48	4000	84.2
48	8000	85.1
56	4000	83.6
56	8000	82.6
64	4000	85.7
64	8000	84.6

There is a tendency of increasingly better results as the block sizes get bigger, at least up to block size 40. This happens because the segmentation process tends to include all relevant motion information from the cases with smaller block sizes into the cases with bigger block sizes. Note though, that this tendency is not a strict rule, as the cases with bigger block sizes are much more sensitive to threshold variations, and thus pose a tougher challenge in tuning the threshold values. This can be more easily identified in Fig. 8. Additionally, by observing the accuracy results to come to these conclusions, we are assuming that these block matching settings have no interaction or confounding with any other parameters of the process of generating and classifying the descriptor.

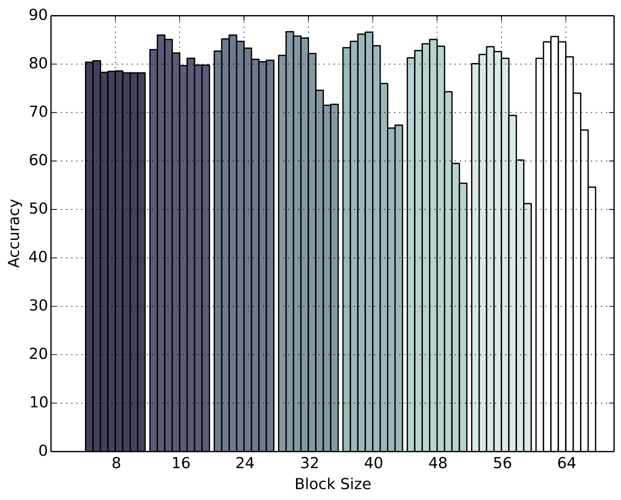


Fig. 8. Bar graph showing classifier accuracy for each parameter setting. The bars are grouped by block size and each bar within a group shows the accuracy value for a different threshold value.

In Fig. 8, each group of bars depicts a fixed block size and the bars within each group show the results for different threshold values, increasing from left to right. On high threshold cases, VSBMA essentially degenerates to BMA, where the blocks may be too big, containing more than one direction of motion. On low threshold cases, the blocks might have become even smaller than the object they are supposed to encapsulate, throughout the segmentation process. Since all the vectors have the same weight in the descriptor computation, regardless of their size, having too many or too few vectors brings back the same problems found with BMA: failure to capture fine motion and sensitivity to noise. By comparing the groups, it is possible to see the difference in behaviour regarding the threshold values. For instance, for block sizes 8 and 16, any threshold value above 8000 can be considered high, producing very similar results amongst themselves.

Fig. 9 shows a contour plot of the same data. In lighter shades we can see the higher accuracy cases. This visualization allows for a quick recognition of what may be the optimal parameter setting, or at least delineate a relation between the parameters in order to achieve good results. It also shows the effect of overestimating threshold values, leading to very poor results, as low as 51.2% accuracy.

State-of-the-art comparison: Although still below the state-of-the-art recognition rates like 93.2% in [10] and 95.0% in [11], the results obtained with VSBMA are somewhat comparable. Especially considering the technique has not been thoroughly explored or optimized for the Human Action Recognition application, and has real time computation capability. Moreover, we note that the best recognition rates from the literature are obtained with the combination of several video characteristics [10], [11]. This often leads to a very demanding process, in terms of computational effort.

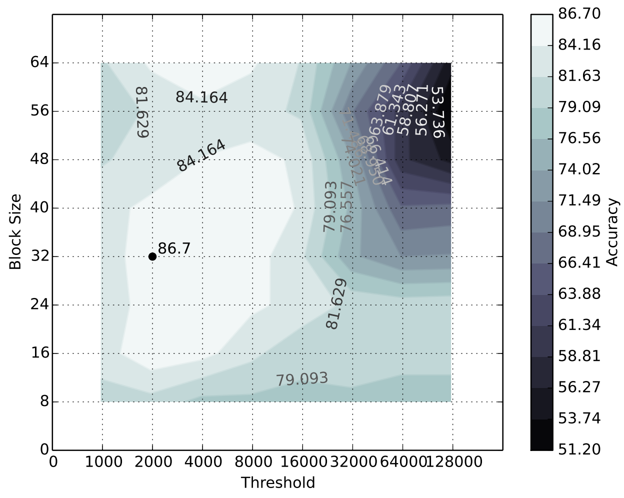


Fig. 9. Contour plot showing classifier accuracy for each parameter setting. Lighter colors indicate higher accuracy values. The highest value is marked as a black dot.

In our work, we use only motion information extracted with VSBMA. Furthermore, in this work we do not explore parameters beyond initial block size and splitting threshold. Throughout descriptor computation and classification, there are other parameters to be tuned, such as descriptor size (related to number of histogram bins), standard deviation σ of histogram Gaussian weighing, other error functions and search strategies, search window size, and minimum block size. Although the results may be sensitive to changes in these parameters, we view them as a sort of fine tuning, so we rely on values found in the literature and previous experience with some of the tools used in this work. We focus mainly on VSBMA and its inherent parameters, for the sake of clarity and to establish a proof of concept that since VSBMA is largely employed in video compression, its output carries valuable motion information that could be used in the action recognition context.

B. Frame Rates

Tables IV and V show summaries of the results for VSBMA regarding execution speed and memory usage, respectively. The fourth column of these tables refers to relative measurement errors, calculated considering a 95% confidence level. To obtain such values, we calculate the size of the confidence interval for said level and divide it by the mean value of the samples. The number of samples for the frame rates is the total number of videos in the dataset, and the number of samples for block counts is the total number of frames in the dataset.

Note that all of the tests show frame rates higher than 25fps, even when considering the error margins. This goes also for the worst case scenarios, with big initial block sizes and low thresholds, that could lead to a lot of segmentation overhead during the process. The fairly low resolution of the videos have to also be accounted for. It contributes to a smaller variation of block sizes in the same frame and to higher frame rates.

TABLE IV
VSBMA RUNNING SPEED RESULTS CONSIDERING DIFFERENT INITIAL BLOCK SIZES AND THRESHOLD VALUES.

Block Size	Threshold	Average FPS	Relative Error
8	1000	144.217	16.07%
8	2000	151.006	10.02%
16	2000	119.511	20.36%
16	4000	139.584	11.23%
24	2000	98.233	26.63%
24	4000	117.803	22.23%
32	2000	74.557	15.35%
32	4000	96.066	25.66%
40	4000	88.897	20.60%
40	8000	108.928	26.14%
48	4000	51.773	10.97%
48	8000	63.885	17.66%
56	4000	45.687	8.53%
56	8000	55.651	11.02%
64	4000	49.460	8.47%
64	8000	59.554	13.05%

Fig. 10 shows a bar graph of the full data for these experiments. Once again the bars are grouped by block size and each individual bar in a group shows the frame rate for a different threshold value, increasing from left to right. The error bars depicted represent the confidence intervals for the samples.

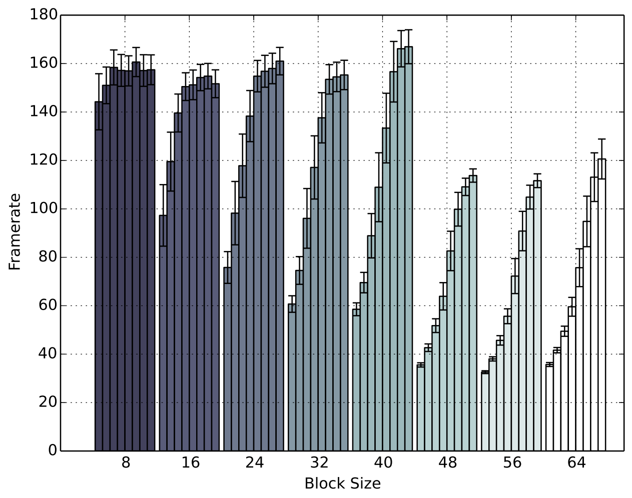


Fig. 10. Bar graph showing running speed for each parameter setting. The bars are grouped by block size and each bar within a group shows the frame rate for a different threshold value. Error bars show the confidence intervals for each case.

Just like the accuracy results, frame rates are more affected by threshold variation on bigger block size cases. However, this time, lower threshold values account for lower frame rates, since the split operation becomes much more frequent than in higher threshold cases. Another aspect that can be observed in this graph is the drop on frame rates for block sizes greater than 40. Although not easily verifiable, this might have to do with the dataset resolution and border handling. In our implementation, the intensity values for pixels beyond image dimensions are computed by demand, whenever they need to be evaluated. For the cases with bigger blocks, there is a greater chance that a portion of a block is out-of-bounds, thus leading to a number of pixel intensities calculations in order to compare two blocks.

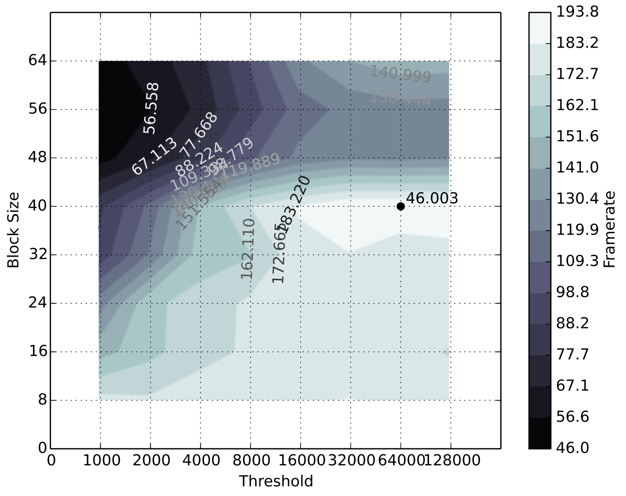


Fig. 11. Contour plot showing running speed for each parameter setting on VSBMA. Lighter colors indicate higher frame rate values.

Fig. 11 shows a contour plot of average frame rates for the same experiments as above. Lighter shades show higher frame rates. This graph highlights one important aspect of the method, that the execution speed is not a trade-off versus recognition accuracy. Note that the best recognition rates were obtained with block sizes 32 and 40, and threshold values of 2000, 4000, and 8000. None of these parameter combinations are within the region of higher frame rates, but they are not within the region of lower frame rates either. In fact, the two contour plots represent very distinct surfaces, with no apparent correlation between them.

C. Block Counts

Table V shows some of the block counts results. As already mentioned, this values reflect memory usage and are also somewhat related to running speed, since the block segmentation process not only increases the number of blocks, but also requires additional memory allocation, function calls, and block comparisons. It is possible to notice a difference in relative error values, when compared to Table IV. As threshold values increase, relative error values drop much more abruptly, especially in smaller block size cases. This can be also seen in Fig. 12.

Fig. 12 shows a bar graph for the block counts of VSBMA. This graph is organized like the previous ones, with bars grouped by block size and individual bars depicting increasing threshold values, from left to right. Note how the error bars, corresponding to the confidence intervals, reduce greatly in size as the threshold values increase. Lower threshold values allow the segmentation of the frames to be much more heterogeneous, whereas higher values lead to no variation at all in the segmentation. In the latter cases, VSBMA degenerates to BMA, where all the frames have the exact same number of blocks. This phenomenon reinforces the idea that the matching errors are much lower than the maximum value attainable and that they may be distributed along a decreasing power law curve.

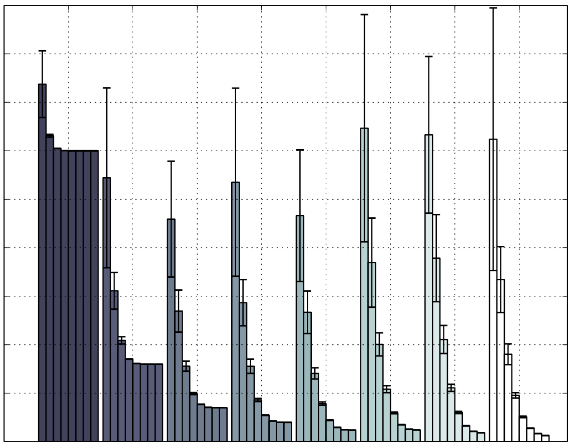


Fig. 12. Bar graph showing block count for each parameter setting. The bars are grouped by block size and each bar within a group shows the block count for a different threshold value. Error bars show the confidence intervals for each case.

Fig. 13 shows a contour plot of block counts experiments. Lighter shades indicate the least amount of blocks. This visualization shows a much more direct and monotonic influence of the parameters over the measurement, block count in this case. Contrary to accuracy or frame rates, block counts have no other parameters or events that affect the values. In the case of accuracies, the process of generating and classifying

TABLE V
VSBMA BLOCK COUNTS. EXPERIMENTS WITH VARIOUS INITIAL BLOCK SIZES AND THRESHOLD VALUES.

Block Size	Threshold	Average Block Count	Relative Error
8	1000	368.713	18.66%
8	2000	315.518	0.95%
8	4000	302.474	0.09%
16	2000	155.602	24.32%
16	4000	104.480	7.17%
16	8000	85.279	0.59%
24	2000	134.677	32.21%
24	4000	77.886	13.29%
24	8000	49.591	4.30%
32	2000	143.315	33.26%
32	4000	77.792	18.90%
32	8000	42.992	7.47%
40	4000	70.426	16.39%
40	8000	39.301	8.46%
40	16000	22.242	3.51%
48	4000	100.364	23.73%
48	8000	54.117	13.28%
48	16000	29.691	6.21%
56	4000	105.414	27.46%
56	8000	55.510	13.24%
56	16000	30.123	7.41%
64	4000	90.178	23.80%
64	8000	47.771	11.58%
64	16000	25.551	6.61%

the descriptor takes into account a number of parameters and operations that could affect the final results. In the case of frame rates, the two examined parameters alter the influence of each other, and some implementation details provide an array of events that could skew the measurements.

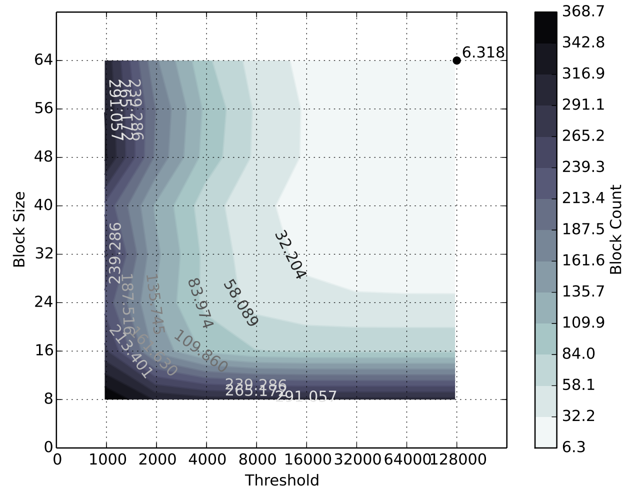


Fig. 13. Contour plot showing block count for each parameter setting. Lighter colors indicate lower number of blocks.

Once again, just like in the frame rate contour plot (Fig. 11), there seems to be no correlation between block count and recognition accuracy, in the sense that no trade-off or direct relation can be observed. The parameter values that lead to the best accuracy results produce some intermediate number of blocks, and the surfaces depicted by Figs. 9 and 13 are clearly distinct.

VI. Conclusion

We presented a tensor self-descriptor obtained from Variable Size Block Matching Algorithm. The displacement vectors computed by VSBMA are represented with histograms which are then coded into orientation tensors.

This is a work that intends to integrate the Block Matching technique into the field of Human Action Recognition. For such purposes, it provides a baseline in terms of parameter exploration. We regard our approach as a promising work, since it yields results close to those of state-of-the-art methods and still has room for a number of improvements. Moreover, it has low complexity in terms of time and space.

Future works may include several improvements, both on VSBMA and on its use for action recognition. Better exploration of the parameters, adaptive threshold values, block merging operations and different block geometry are a few examples of improvements that can be made on VSBMA. As for human action recognition, the integration of VSBMA and other, more complex, techniques and datasets is going to be the next improvement of this work. Datasets containing multiple actors and background actions per video pose an important challenge to validate our descriptor since they are a better representation of realistic scenarios.

Acknowledgment

Authors would like to thank FAPEMIG, CAPES, CNPq and UFJF for funding.

References

[1] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. COM-29, No 12, pp. 1799–1808, 1981. doi: 10.1109/TCOM.1981.1094950

[2] V. F. Mota, E. d. A. Perez, M. B. Vieira, L. Maciel, F. Precioso, and P. H. Gosselin, "A tensor based on optical flow for global description of motion in videos," in *Conference on Graphics, Patterns and Images (SIBGRAPI).* IEEE, 2012. doi: 10.1109/SIBGRAPI.2012.48 pp. 298–301.

[3] A. Kläser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3d-gradients," in *British Machine Vision Conference*, sep 2008. doi: 10.1.1.165.3934 pp. 995–1004. [Online]. Available: http://lear.inrialpes.fr/pubs/2008/KMS08

[4] A. M. Amel, B. A. Abdessalem, and M. Abdellatif, "Video shot boundary detection using motion activity descriptor," *Journal of Telecommunications*, vol. 2, no. 1, pp. 54–59, 2010.

[5] A. Hafiane, K. Palaniappan, and G. Seetharaman, "Uav-video registration using block-based features," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 2, 2008. doi: 10.1109/IGARSS.2008.4779192 pp. 1104–1107.

[6] A. Puri, H. Hang, and D. Schilling, "Interframe coding with variable block-size motion compensation," in *IEEE Global Telecommunication Conference*, 1987. doi: 10.1.1.23.7217 pp. 65–69.

[7] M. Chan, Y. Yu, and A. Constantinides, "Variable size block matching motion compensation with applications to video coding," in *IEEE inproceedings 1990*, vol. 137, No.4, 1990. doi: 10.1049/ip-i-2.1990.0029 pp. 205–212. [Online]. Available: http://dl.acm.org/citation.cfm?id=646271.685624

[8] I. R. K. P. Muralidhar, C.B. Rama Rao, "Efficient architecture for variable block size motion estimation of h.264 video encoder," *International Conference on Solid-State and Integrated Circuit (ICSIC)*, vol. 32, p. 6, 2012. doi: 01.IJSIP.5.1.1

[9] Y. Ji, A. Shimada, and R.-i. Taniguchi, "A compact 3d descriptor in roi for human action recognition," in *IEEE TENCON*, 2010, pp. 454–459.

[10] V. F. Mota, E. d. 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 Recognition Letters*, vol. 31, pp. 85–91, 2013. doi: 10.1016/j.patrec.2013.08.008

[11] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013. doi: 10.1007/s11263-012-0594-8

[12] S. L. Horowitz and T. Pavlidis, "Picture segmentation by a tree traversal algorithm," *J. ACM*, vol. 23, no. 2, pp. 368–388, Apr. 1976. doi: 10.1145/321941.321956

[13] I. Rhee, G. R. Martin, S. Muthukrishnan, and R. A. Packwood, "Quadtree-structured variable-size block-matching motion estimation with minimal error," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, No. 1, pp. 42–50, 2000. doi: 10.1109/76.825857

[14] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, No. 4, pp. 438–442, 1994.

[15] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems For Video Technology*, vol. 6, No. 3, pp. 313–317, 1996. doi: 10.1.1.29.5956

[16] J. Lu and M. L. Liou, "A simple and efficient search algorithm for block-matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, No. 2, pp. 429–433, 1997. doi: 10.1109/76.564122

[17] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, No. 2, pp. 287–290, 2000. doi: 10.1.1.330.3947

[18] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1442–1449, 2002. doi: 10.1109/TIP.2002.806251

[19] A. Barjatya, "Block matching algorithms for motion estimation," *IEEE Transactions Evolution Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[20] J. W. Kim and S. U. Lee, "Hierarchical variable block size motion estimation technique for motion sequence coding," *Optical Engineering*, vol. 33, no. 8, pp. 2553–2561, 1994. doi: 10.1.1.23.261

[21] B. Johansson and G. Farnebäck, "A theoretical comparison of different orientation tensors," in *Proceedings of the SSAB Symposium on Image Analysis*, 2002. doi: 10.1.1.8.6041 pp. 69–73.

[22] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local svm approach," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, vol. 3. IEEE, 2004. doi: 10.1109/ICPR.2004.1334462 pp. 32–36.

[23] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, no. 12, pp. 1649–1668, Dec 2012. doi: 10.1109/TCSVT.2012.2221191

Fábio Luiz Marinho de Oliveira: Obtained a graduate degree in Computer Science from Universidade Federal de Juiz de Fora in 2014. He is currently a masters student in the Computer Science post-graduate program (PGCC) at the same university. He has been also a member of the Group for Computer Graphics, Image and Vision (GCG) since 2011. His main areas of interest are Computer Graphics, Image Processing, Computer Vision and Machine Learning, specifically video human action recognition.

Helena de Almeida Maia: Helena de Almeida Maia is a M.Sc. student in the Computer Science post-graduate program (PGCC) at Universidade Federal de Juiz de Fora. She has been a member of the Group for Computer Graphics, Image and Vision (GCG) since 2012. She obtained a graduate degree in Computer Science from Universidade Federal de Juiz de Fora in 2014. Her main interest areas include Computer Graphics, Computer Vision, especially Smart Surveillance researches, Image Processing and Machine Learning.

Virgínia Fernandes Mota: Virginia Fernandes Mota is a Ph.D. student in the Computer Science Department at Universidade Federal de Minas Gerais, Brazil. She is currently a researcher in the Digital Image Processing Group (NPDI) supervised by Prof. Arnaldo de Albuquerque Araújo and Prof. Marcelo Bernardes Vieira. She holds a M.Sc. Degree in Computer Science (2009-2011) at Federal University of Juiz de Fora, Brazil, and a Master in Systèmes Intelligents et Communicants (2009-2010) at École Nationale Supérieure de l'Electronique et de ses Applications, France. She also holds B.Sc. degree in Computer Science (2005-2008) at Universidade Federal de Juiz de Fora, Brazil. She was a temporary teacher in the Computer Science Department at Universidade Federal de Minas Gerais (2012). She is a professor at the Informatics Department of COLTEC-UFMG.

Marcelo Bernardes Vieira: 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 and 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 was researcher at Instituto de Matemática Pura e Aplicada between 2003 and 2005. 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 modelling, 3D reconstruction and pattern recognition. He is coordinator of the Group for Computer Graphics, Image and Vision at UFJF.

Arnaldo de Albuquerque Araújo: Arnaldo de Albuquerque Araújo has a Ph.D. in Electrical Engineering Sciences from UFPB (1987). Post-graduated in Image Processing at RWTH Aachen (1981-1985). Post-doc in informatics from ESIEE Paris (1994-1995), ENSEA Cergy-Pontoise (2005) and UPMC Paris 6 (2008-2009). He is currently an Adjunct Professor at DCC/UFMG (since 1990) and coordinates the Digital Image Processing Center NPDI/DCC/UFMG. He was also an Adjunct Professor at DEE/CCT/UFPB (1978-1989). His current research topics belong to image processing, computer vision, multimedia information systems, and visual information retrieval.