Variable Size Block Matching Trajectories for Human Action Recognition

Fábio L.M. de Oliveira(✉) and Marcelo B. Vieira

Universidade Federal de Juiz de Fora, Juiz de Fora, MG, Brazil
{fabio,marcelo.bernardes}@ice.ufjf.br

Abstract. In the context of the human action recognition problem, we propose a tensor descriptor based on sparse trajectories extracted via Variable Size Block Matching. Compared to other action recognition descriptors, our method runs fast and yields a compact descriptor, due to its simplicity and the coarse representation of movement provided by block matching. We validate our method using the KTH dataset, showing improvements over a previous block matching based descriptor. The recognition rates are comparable to those of state-of-the-art methods with the additional feature of having frame rates close to real-time computation.

Keywords: Human action recognition · Variable size block matching · Self-descriptor · Tensor descriptor

1 Introduction

Human Action Recognition (HAR) has been a rapid growing research field over recent years. Along with other computer vision themes, it has been boosted especially by the massive popularization of video recording devices, such as phone cameras and surveillance equipment. These devices generate an amount of video data too big to have its semantic value or significance examined by people. Hence the need of an automated system capable of extracting this information from videos.

Many, if not all, of the methods for this purpose rely on detecting and tracking movement in sequences of frames. That is the case with optical flow, 3-D gradients, spatio-temporal interest points, block matching, amongst others. The assumption is that movement can be detected, described, and used to categorize different actions portrayed in a video.

In the case of block matching, which is the focus of this paper, this motion information has been largely used for video compression since the introduction of the method by Jain and Jain [9]. It consists in dividing a frame into rectangular blocks and searching for corresponding blocks on the following frame. These corresponding blocks are the ones that are most similar regarding brightness or color. This results in a set of displacement vectors, one for each block. The original authors [9] described it as a "piecewise translation".

O. Gervasi et al. (Eds.): ICCSA 2015, Part I, LNCS 9155, pp. 283–297, 2015.
DOI: 10.1007/978-3-319-21404-7_21

In this work, we propose a video tensor descriptor based on trajectories obtained via multiple variable size block matchings [26]. This work is an improvement over our latest works [6]. Compared to other well known action recognition descriptors, like ones based on dense trajectories or 3-D gradients, our method is fast and yields a compact descriptor, due to its coarser representation of the movement. In fact, the method runs fast enough to be potentially incorporated into a real-time application and the final descriptor file size for the whole dataset rarely exceed that of a single video.

The following subsections show related works and an overview of the method. In Sec. 2 we go into details for each stage of the method. Section 3 shows the experimental environment and Sec. 4 the results and discussion.

1.1 Related Work

Since block matching was introduced by Jain and Jain [9] and the later introduction of its variable block size variant [3,26], it has appeared in a number of publications concerning video coding [2,3,5,13,24,29] and was part of specifications of codecs like H.263 [15] and H.264 [4,23], amongst others. In this work, we employ block matching as a motion flow under the assumption that it generates quality and compact descriptors, as it has been so broadly explored in the video coding and compression contexts.

Aside from video compression, block matching has been used for shot boundaries detection by Amel et al. [1], and video registration by Hafiane et al. [7].

As a crucial component of the block matching routine, several search strategies have been proposed in order to speed up the process of finding the best matches. Some examples include: Three Step Search [16], Four Step Search (4SS) [25], Simple and Efficient Search [18], Diamond Search [34], and others. In our implementation, we chose 4SS as it is a simple to implement, steepest descent based strategy that runs approximately 10 times faster than the exhaustive approach.

In the action recognition context, techniques like optical flow [21] and 3D gradients [14] have been shown to produce high recognition rates. The same applies for the use of trajectories. Both Wang et al. [31,33] and Jain et al. [10] combine dense sampled trajectories and other descriptors, like Motion Boundary Histogram (MBH) and Histogram of Optical Flow (HOF) to achieve very high recognition rates on several datasets. In this work, we use only block matching information to generate the descriptor. With this, we obtain a computationally inexpensive descriptor, both in terms of time and space, with recognition rates comparable to state-of-the-art.

To condense the motion information extracted, a histogram is commonly used [11,14,20,32]. Histograms are interesting for video description as they are simple structures which encode a compact representation of the motion information. In Mota et al. [20], 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 uncertainties of motion, and thus, could carry more useful information than a

histogram. In this work, the final video descriptor is an orientation tensor that accumulates information from all trajectories.

1.2 Overview

The general workflow of a human action recognition system consists of three main tasks: motion extraction, motion representation, and action classification. Our focus is on the first two tasks, that is, computing a descriptor from block matching information. A modified Variable Size Block Matching Algorithm (VSBMA) [26] extracts trajectories from a set of frames, and thus is responsible for the first task. For the motion representation task, the vectors composing these trajectories are accumulated into a histogram of directions and then an orientation tensor is coded from it. These steps are repeated for all the frames of a video, in order to generate the video descriptor. Figure 1 illustrates this sequence of steps.

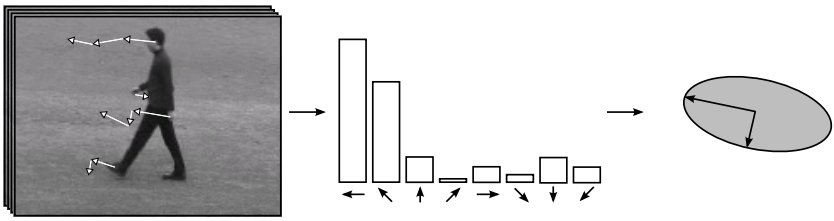


Fig. 1. Technique overview. From left to right: trajectories generated from the block matching routine, histogram of directions, and orientation tensor. The ellipse is merely an illustration since the tensor dimension is greater than 2.

2 Proposed Method

2.1 Variable Size Block Matching

The main algorithm consists in dividing a so called "reference frame" into blocks of pixels and finding for each one a corresponding block in a so called "target frame" which minimizes a dissimilarity (or error) function. For each block, the algorithm outputs a displacement vector between the coordinates of a reference block and its corresponding target.

This dissimilarity function is generally based on pixel intensities within the analyzed blocks. Although others can be found in related literature [3,9,13], In our implementation the function of choice was the Sum of Absolute Differences (SAD). In previous works and preliminary tests, we found that employing other error functions such as Mean Absolute Differences (MAD) or Sum of Squared Differences (SSD) had very slight or no impact at all regarding the action recognition context.

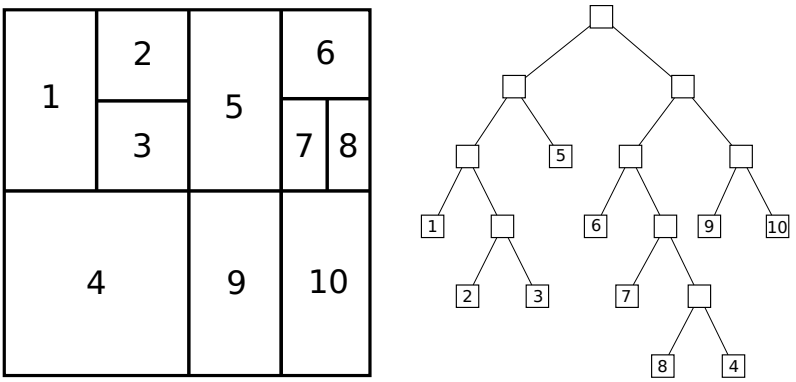


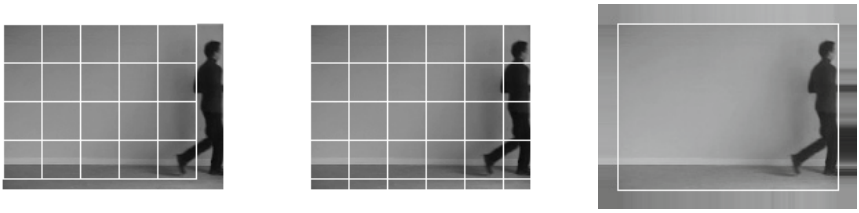
Fig. 2. Binary tree image segmentation from [3]. (a) Frame segmentation scheme. (b) Tree structure corresponding to (a).

When matching blocks, the blocks evaluated in the target frame are restricted to a close vicinity of the block coordinates in the reference frame. This vicinity is called a search window. The search window is established under the assumption that the movement of objects between frames is somewhat smooth, so it is not necessary to search the whole target frame. This greatly reduces the cost of the search, especially for higher resolution sequences. Note also that the size of the search window limits the magnitude of the displacement vectors. 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 $(\pm 7, \pm 7)$.

Another noteworthy element of the algorithm is the search strategy. Even with a search window, evaluating all blocks within it is still too big an effort. To reduce this cost, a number of strategies have been proposed over the years [16, 18, 22, 25, 34]. For this work, we chose Four Step Search (4SS) [25] as the search strategy. 4SS is a steepest descent strategy that compares at most 27 blocks to find the match, as opposed to evaluating all the blocks of the search window, which requires 225 comparisons.

All these elements are also present in a conventional Block Matching Algorithm (BMA). What differentiates VSBMA from BMA is that the sizes of the blocks in VSBMA change during the matching routine. All blocks have an initial size, but blocks that have a minimum matching error above a fixed error threshold are divided into smaller blocks and matched again. This process repeats until the error is below the threshold or the size of the blocks reach a fixed minimum size of 4×4 pixels. Just like first proposed in [3], we split the blocks into two smaller blocks, alternating between horizontal and vertical partitions. This is appropriately represented by a binary tree (Fig. 2), in which the leaves are blocks of varying sizes. As in picture segmentation [8], the goal is to have blocks that encompass homogeneous regions, possibly representing objects in the scene.

Fig. 3. VSBMA displacement vectors. Hotter colors indicate bigger blocks.



(a) (b) (c)

Fig. 4. Treatment for out-of-bounds block coordinates. (a) No treatment. Blocks are marked in white and regions close to the right and bottom of the image are not considered. (b) Fill solution. Gaps in said regions are occupied with blocks with different sizes. (c) Frame extension. The white rectangle now highlights the original frame. Intensity values for pixels outside of this rectangle are the same as those in the border.

Figure 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.

Boundary Treatment and Trajectory Extration. Each video frame 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, and thus, are not considered, as in Fig. 4. In this case, a border treatment becomes necessary to completely cover a frame. To solve this

issue we use the initial segmentation depicted in Fig. 4, positioning as many whole blocks as possible within the frame and then completing the remainder of the frame with rectangular blocks, with initial sizes different from other blocks.

Even with this approach of filling the frame with blocks, another issue remains. Since all blocks are confined within the bounds of the frame, objects leaving the scene result in extraneous vectors with high error values. The frame is then extended as shown in Fig. 4 to overcome this problem. The intensity value for an out-of-bounds pixel is the same as its closest neighbor in the borders of the image, creating a stretching effect. Note that this solution also supplants the previous one, as blocks that do not fit in the original frame, fit in the extended frame. This is also coherent with block matching implementations specified in H.263 and H.264 (HEVC) [15,30].

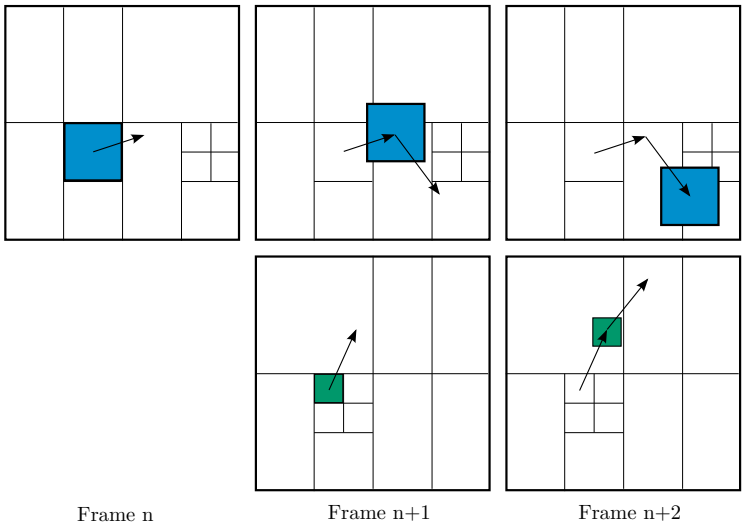


Fig. 5. Example trajectories starting on frames n (top) and $n+1$ (bottom). On the upper sequence, an object is tracked from frame n to frame $n+2$. Segmentation remains as on frame n. On the lower sequence, another object is tracked from frame $n+1$ to frame $n+3$ (not depicted). In this case, the segmentation used is the one resulting from matches between frames $n+1$ and $n+2$.

In order to extract trajectories, a set of frames, instead of just a pair, is analyzed. When doing so, VSBMA is used between the first and second frames of the set, and the following matches are carried out as conventional block matching, using the target of a previous match as reference for the following one. This way, the frame segmentation occurs only once and stays the same for the rest of the round of computations. This procedure is repeated in such way that all frames of the video serve as starting point for a trajectory. Figure 5 shows trajectories starting on two consecutive frames presenting different behaviours. Notice that

the same region of the frame may be segmented in two different ways, depending on the reference frame, resulting in different trajectories. This is the case when objects appear or disappear during the sequence, enter or leave the camera's field of view, or if the sequence has distinctive shots or cuts. For such cases, especially regarding scene cuts, the trajectories carry erroneous motion information, since the assumption of continuous motion is toppled. By overlapping multiple trajectories, these errors have less impact on the overall accumulated motion information.

The use of these sparse trajectories computed via block matching is an attempt to obtain quality results, as found in other trajectory based works [10, 31,32], but at a lower computational cost. Not only block matching serves as a low-cost kind of flow, compared to gradients or optical flow, but it also provides a coarser representation of objects in the scene compared to dense or keypoint sampling. This results in less trajectories carrying possibly the same meaningful motion information as a set of dense trajectories.

2.2 Generating the Descriptor

Histogram of Directions. The result of block matching 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 $r = \parallel d(i,j) \parallel$. All displacement vectors of all trajectories starting in the reference frame are used to form a histogram of directions.

A motion direction histogram is used as a compact representation of the motion vector field obtained from each frame. It is defined as the column vector $\boldsymbol{h}_f = (h_1, h_2, \ldots, h_{n_\theta})^T$, 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.

Consequently, a histogram is computed for each video frame, represented by a vector $\boldsymbol{h}_f$ with n_θ elements. It encodes all displacements of all blocks forming trajectories starting at the frame and spreading throughout a fixed number of frames ahead.

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 [12]. Given a vector $\boldsymbol{v} \in \mathbb{R}^n$, it can be represented by the tensor $\mathbf{T} = \boldsymbol{v}\boldsymbol{v}^{\mathrm{T}}$. Then, we use the orientation tensor to represent the histogram $\boldsymbol{h}_f \in \mathbb{R}^{n_\theta}$. The frame tensor, $\mathbf{T}_f \in \mathbb{R}^{n_\theta \times n_\theta}$, is given by:

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

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

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

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

using all 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 motions captured in the histograms are too different from each other, we obtain an isotropic tensor which does not hold useful information. But, if the accumulation results in an anisotropic tensor, it carries meaningful average motion information of the frame sequence [20].

3 Experiments

The method is validated using the KTH dataset [28], 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 4 different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes, and indoors. In this work, contrary to the protocol suggested in [28], the dataset is not split in sequences. The videos have a resolution of 160×120 pixels and 25fps frame rate.

We use a SVM classifier to evaluate our descriptor on KTH. All tests were run on an Intel®Core™2 Quad Q9550 2.83GHz with 4GB memory running a single thread per video. This is an important remark, as the speed of the method could be even further improved running more parallel threads or in an up to date system.

Table 1 shows the parameter values used on our experiments. 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. However, we chose to focus on the three main parameters: initial square block width, VSBM error threshold and trajectory size. We have conducted experiments with 7 initial block sizes, 4 threshold values, and 7 trajectory sizes, to a total of $7 \times 4 \times 7 = 196$ different parameter settings. For all the other parameters, we rely on values found in the literature and previous experience with some of the tools used in this work.

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 [30], and recognition rates from preliminary experiments that

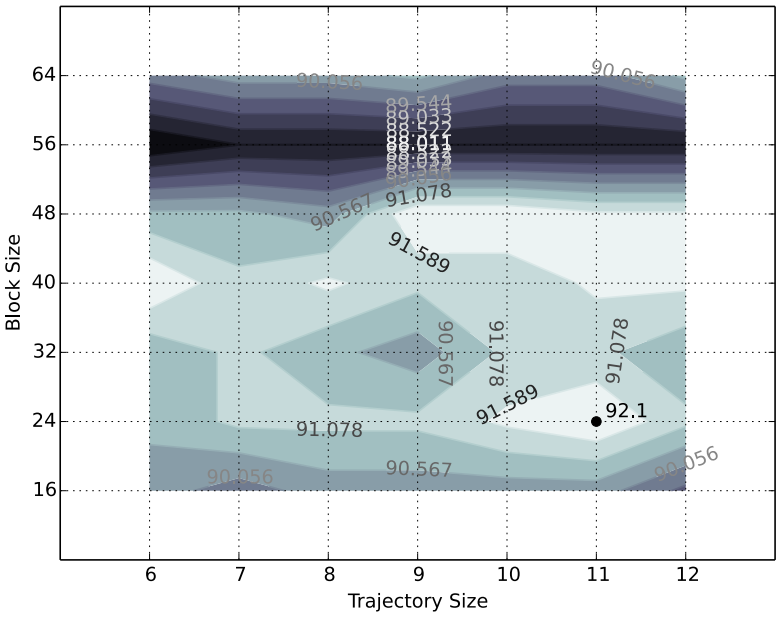


Fig. 6. Contour plot showing classifier accuracy for each parameter setting regarding block and trajectory sizes. Lighter colors indicate higher accuracy values. The highest value is marked as a black dot.

Table 1. Parameter values for experiments with VSBMA

Parameter	Values
Initial block width (pixels)	16, 24, 32, 40, 48, 56, 64
VSBM error threshold	2000, 4000, 8000, 16000
Trajectory size (frames)	6, 7, 8, 9, 10, 11, 12

showed a decrease in quality for larger blocks. As for threshold values, the choice was based on the maximum matching 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.

To assess and compare the quality of our descriptor in the action recognition context, we measure the recognition rate, which is the output of a SVM classifier, which takes the descriptors for the whole database. We follow the same classification protocol as [28]. The classifier produces 6 recognition rates for each 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

Table 2. Summary of best recognition rates

Block Size	Threshold	Trajectory Size	Accuracy
24	2000	10	91.7
24	2000	11	92.1
40	4000	8	91.7
40	4000	9	91.2
40	4000	11	91.7
40	8000	6	92.1
40	8000	8	91.7
40	8000	12	91.7
48	8000	9	92.1
48	8000	10	92.1
48	8000	11	91.7
48	8000	12	91.7

Sec. 4. In order to evaluate the efficiency of the method, we consider the frame rate at which block matching operates.

4 Results and Discussion

4.1 Recognition Rates

Table 2 shows a summary of the results obtained. Notice the predominance of block sizes 40 and 48, and of thresholds 4000 and 8000. This can be attributed to the segmentation process including all relevant motion information from the cases with smaller block sizes into the cases with bigger block sizes. This is also true for the cases with block size 56 and 64, except that these block sizes do not produce better results. In these cases, the initial segmentation includes too many of out-of-bounds pixels. Consider, for example, a block size of 56 and the dataset resolution of 160×120. Horizontally, 3 blocks fit within the image with only 2 columns of pixels out-of-bounds, but vertically only 2 blocks fit perfectly, and the third one has 48 rows of out-of-bounds pixels. Although not amongst the best results, the recognition rates of these parameter settings reach upwards of 90.7%.

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.

Figure 6 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 previously mentioned, where block size 56 has noticeably poorer results.

State-of-the-art comparison: We can see a clear improvement over our previous block matching based descriptor [6], especially considering that the best result

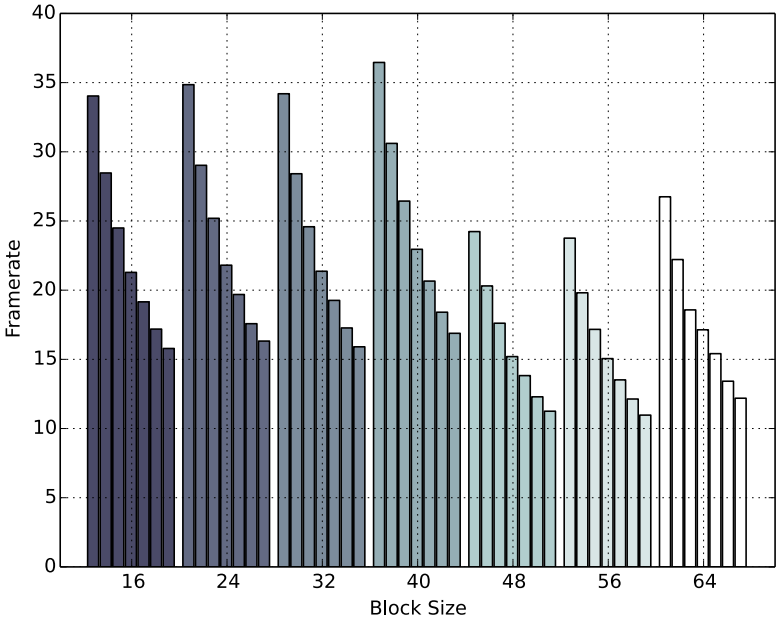


Fig. 7. Bar graph showing running speed for each parameter setting. Each group of bars represents a block size and bars within a group represent a different trajectory size, increasing from left to right.

Table 3. Comparison with state-of-the-art and previous works for KTH dataset

Authors	Recognition Rate
Klaser et al. (2008) [14]	91.0
Liu et al. (2009) [17]	93.8
Mota et al. (2013) [19]	93.2
Sad et al. (2013) [27]	93.3
Wang et al. (2013) [33]	**95.3**
Figueiredo et al. (2014) [6]	87.7
This work	**92.1**

in that work was obtained with the most computationally expensive method proposed, MSMV. Although still below the state-of-the-art recognition rates like 93.2% in [20] and 95.0% in [33], the results obtained using block matching are comparable. The best recognition rates from the literature are obtained using combinations of video characteristics [10,20,32]. This often leads to a very demanding process, in terms of computational effort. In our work, we use only

motion information extracted with VSBMA, achieving real time computation capability in some parameter settings.

4.2 Frame Rates

Figure 7 shows a bar graph of the frame rates measured in our experiments. Bars are grouped by block size and each bar within a group shows the frame rate for a different trajectory size, increasing from left to right.

By comparing different groups, we can see that the impact of the trajectory size is approximately the same on all cases, reducing frame rates in roughly 50% between trajectory sizes 6 and 12. Like the accuracy results, frame rates are noticeably lower for block sizes greater than 40. 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, greater portions of blocks are out-of-bounds, thus leading to an increased number of pixel intensities calculations in order to compare two blocks.

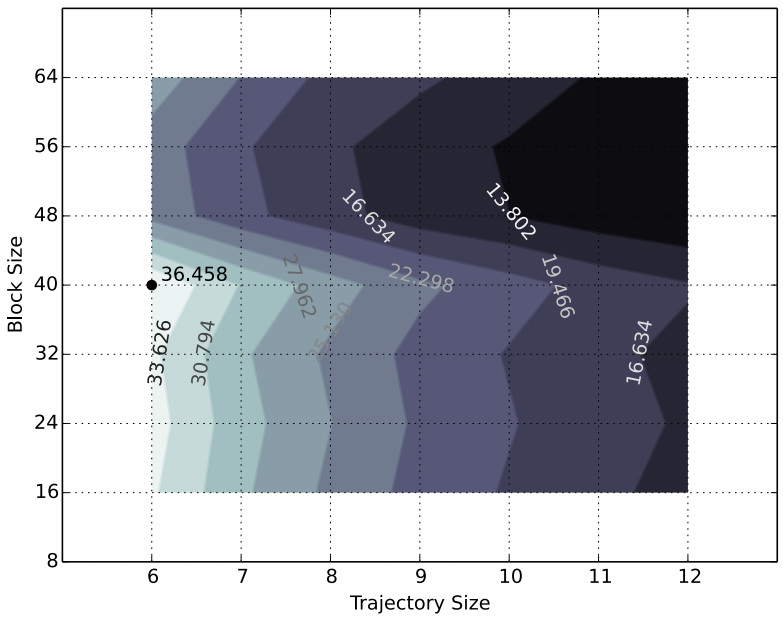


Fig. 8. Contour plot showing running speed for each parameter setting on VSBMA. Lighter colors indicate higher framerate values.

Figure 8 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: the execution speed is not a trade-off versus recognition accuracy. Note that the best recognition rates were obtained with block sizes 40 and 48, and trajectory sizes of 6, 9, 10, and 11. With the exception of the case with block size 40 and trajectory size 6, none of these parameter combinations are within the region of highest 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.

5 Conclusion

We presented a tensor self-descriptor based on variable block matching trajectories. The trajectories are accumulated into orientation histograms which in turn are coded into orientation tensors.

This is a work within the context of Human Action Recognition. It aims to provide a baseline for further developments regarding sparse trajectories and block matching in this context. We view our approach as a promising work, since it yields results close to those of state-of-the-art methods at low computational costs.

Future works may include several improvements, both on block matching and on its use for action recognition. Better exploration of the parameters, adaptive threshold values, adaptive trajectory sizes, block merging operations, and different block geometry are a few examples of improvements that can be made on the block matching end. As for human action recognition, the integration of other, more complex, techniques and datasets is going to be the next improvement of this work.

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

References

1. Amel, A.M., Abdessalem, B.A., Abdellatif, M.: Video shot boundary detection using motion activity descriptor. Journal of Telecommunications **2**(1), 54–59 (2010)
2. Calzone, S., Chen, K., Chuang, C.C., Divakaran, A., Dube, S., Hurd, L., Kari, J., Liang, G., Lin, F.H., Muller, J., Rising, H.K.: Video compression by mean-corrected motion compensation of partial quadtrees. IEEE Transactions on Circuits and Systems for Video Technology **7**(1), 86–96 (1997)
3. Chan, M., Yu, Y., Constantinides, A.: Variable size block matching motion compensation with applications to video coding. IEEE in Proceedings **137**(4), 205–212 (1990). http://dl.acm.org/citation.cfm?id=646271.685624
4. Chen, C.Y., Chien, S.Y., Huang, Y.W., Chen, T.C., Wang, T.C., Chen, L.G.: Analysis and architecture design of variable block-size motion estimation for h.264/avc. IEEE Transactions on Circuits and Systems I: Regular Papers **53**(3), 578–593 (2006)

5. Choi, S.J., Woods, J.: Motion-compensated 3-d subband coding of video. IEEE Transactions on Image Processing **8**(2), 155–167 (1999)
6. Figueiredo, A.M.O., Maia, H.A., Oliveira, F.L.M., Mota, V.F., Vieira, M.B.: A video tensor self-descriptor based on block matching. In: Murgante, B., Misra, S., Rocha, A.M.A.C., Torre, C., Rocha, J.G., Falcão, M.I., Taniar, D., Apduhan, B.O., Gervasi, O. (eds.) ICCSA 2014, Part VI. LNCS, vol. 8584, pp. 401–414. Springer, Heidelberg (2014)
7. Hafiane, A., Palaniappan, K., Seetharaman, G.: Uav-video registration using block-based features. In: IEEE International Geoscience and Remote Sensing Symposium (IGARSS), vol. 2, pp. 1104–1107 (2008)
8. Horowitz, S.L., Pavlidis, T.: Picture segmentation by a tree traversal algorithm. J. ACM **23**(2), 368–388 (1976). http://doi.acm.org/10.1145/321941.321956
9. Jain, J.R., Jain, A.K.: Displacement measurement and its application in interframe image coding. IEEE Transactions on Communications COM **29**(12), 1799–1808 (1981)
10. Jain, M., Jegou, H., Bouthemy, P.: Better exploiting motion for better action recognition. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2555–2562, June 2013
11. Ji, Y., Shimada, A., Taniguchi, R.I.: A compact 3d descriptor in roi for human action recognition. In: IEEE TENCON, pp. 454–459 (2010)
12. Johansson, B., Farnebäck, G.: A theoretical comparison of different orientation tensors. In: Proceedings of the SSAB Symposium on Image Analysis, pp. 69–73 (2002)
13. Kim, J.W., Lee, S.U.: Hierarchical variable block size motion estimation technique for motion sequence coding. Optical Engineering **33**(8), 2553–2561 (1994)
14. Kläser, A., Marszałek, M., Schmid, C.: A spatio-temporal descriptor based on 3d-gradients. In: British Machine Vision Conference (BMVC), pp. 995–1004, September 2008
15. Ku, C.W., Lin, G.S., Chen, L.G., Lee, Y.P.: Architecture design of motion estimation for itu-t h.263, vol. 3024, pp. 482–493 (1997). http://dx.doi.org/10.1117/12.263260
16. Li, R., Zeng, B., Liou, M.L.: A new three-step search algorithm for block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology **4**(4), 438–442 (1994)
17. Liu, J., Luo, J., Shah, M.: Recognizing realistic actions from videos in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1996–2003. IEEE (2009)
18. Lu, J., Liou, M.L.: A simple and efficient search algorithm for block-matching motion estimation. IEEE Transactions on Circuits and Systems for Video Technology **7**(2), 429–433 (1997)
19. Mota, V.F., Souza, J.I., de A. Araújo, A., Vieira, M.B.: Combining orientation tensors for human action recognition. In: Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 328–333. IEEE (2013)
20. Mota, V.F., Perez, E.D.A., Maciel, L.M., Vieira, M.B., Gosselin, P.H.: A tensor motion descriptor based on histograms of gradients and optical flow. Pattern Recognition Letters **31**, 85–91 (2013)
21. Mota, V.F., Perez, E.D.A., Vieira, M.B., Maciel, L., Precioso, F., Gosselin, P.H.: A tensor based on optical flow for global description of motion in videos. In: Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 298–301. IEEE (2012)
22. Nie, Y., Ma, K.K.: Adaptive rood pattern search for fast block-matching motion estimation. IEEE Transactions on Image Processing **11**(12), 1442–1449 (2002)

23. Muralidhar, P., Rama Rao, C.B., Ranjith Kumar, I.: Efficient architecture for variable block size motion estimation of h.264 video encoder. In: International Conference on Solid-State and Integrated Circuit (ICSIC), vol. 32, p. 6 (2012)
24. Pirsch, P., Demassieux, N., Gehrke, W.: Vlsi architectures for video compression-a survey. Proceedings of the IEEE **83**(2), 220–246 (1995)
25. Po, L.M., Ma, W.C.: A novel four-step search algorithm for fast block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology **6**(3), 313–317 (1996)
26. Puri, A., Hang, H., Schilling, D.: Interframe coding with variable block-size motion compensation. In: IEEE Global Telecommunication Conference, pp. 65–69 (1987)
27. Sad, D., Mota, V.F., Maciel, L.M., Vieira, M.B., Araújo, A.d.A.: A tensor motion descriptor based on multiple gradient estimators. In: Conference on Graphics, Patterns and Images (SIBGRAPI), pp. 70–74. IEEE (2013)
28. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: Proceedings of the 17th International Conference on Pattern Recognition (ICPR), vol. 3, pp. 32–36. IEEE (2004)
29. Sullivan, G., Baker, R.: Rate-distortion optimized motion compensation for video compression using fixed or variable size blocks. In: Global Telecommunications Conference, GLOBECOM 1991. 'Countdown to the New Millennium. Featuring a Mini-Theme on: Personal Communications Services, vol. 1, pp. 85–90, December 1991
30. Sullivan, G., Ohm, J., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (hevc) standard. IEEE Transactions on Circuits and Systems for Video Technology **22**(12), 1649–1668 (2012)
31. Wang, H., Klaser, A., Schmid, C., Liu, C.L.: Action recognition by dense trajectories. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3169–3176, June 2011
32. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Dense trajectories and motion boundary descriptors for action recognition. International Journal of Computer Vision **103**(1), 60–79 (2013)
33. Wang, H., Schmid, C., et al.: Action recognition with improved trajectories. In: International Conference on Computer Vision (2013)
34. Zhu, S., Ma, K.K.: A new diamond search algorithm for fast block-matching motion estimation. IEEE Transactions on Image Processing **9**(2), 287–290 (2000)