UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ana Mara de Oliveira Figueiredo

A Video Self-descriptor based on Sparse Trajectory Clustering

Juiz de Fora

2015

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
INSTITUTO DE CIÊNCIAS EXATAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Ana Mara de Oliveira Figueiredo

A Video Self-descriptor based on Sparse Trajectory Clustering

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Marcelo Bernardes Vieira
Coorientador: Rodrigo Luis de Souza da Silva

Juiz de Fora
2015

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Figueiredo, Ana Mara de Oliveira.
A Video Self-descriptor based on Sparse Trajectory Clustering / Ana Mara de Oliveira Figueiredo. -- 2015.
60 f.

Orientador: Marcelo Bernardes Vieira
Coorientador: Rodrigo Luis de Souza da Silva
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2015.

1. Block Matching. 2. Human action recognition. 3. Self-descriptor. 4. Sparse and dense trajectories. 5. Trajectory clustering. I. Vieira, Marcelo Bernardes, orient. II. Silva, Rodrigo Luis de Souza da, coorient. III. Título.

Ana Mara de Oliveira Figueiredo

A Video Self-descriptor based on Sparse Trajectory Clustering

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Ciências Exatas da Universidade Federal de Juiz de Fora como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Aprovada em 10 de Setembro de 2015.

BANCA EXAMINADORA

Prof. D.Sc. Marcelo Bernardes Vieira - Orientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Rodrigo Luis de Souza da Silva- Coorientador
Universidade Federal de Juiz de Fora

Prof. D.Sc. Alex Fernandes da Veiga Machado
Instituto Federal do Sudeste de Minas Gerais

Prof. D.Sc. Raul Fonseca Neto
Universidade Federal de Juiz de Fora

For my parents, Geraldo and Vera, and my brothers Alan and Lucimara.

ACKNOWLEDGMENTS

Firstly I thank God for giving me strength in this journey. I thank my parents, Geraldo and Vera, for the immense support and encouragement along this trajectory. I thank my boyfriend, Rafael, for the love and understanding in all the moments.

I thank my advisor, Marcelo Bernardes Vieira, and my coadvisor, Rodrigo Luis de Souza da Silva. Thanks to Marcelo Caniato, Helena Maia, Fábio Luiz, Felipe Caetano and the other GCG members for the support in this work and the friendship, especially in difficult times. I thank to my friends, Camila Campos and Alessandra Cristina, for everything, especially for the friendship. Finally, I thank everyone who contributed in some way for my trajectory.

"A sabedoria oferece proteção, como o faz o dinheiro, mas a vantagem do conhecimento é esta: a sabedoria preserva a vida de quem a possui." Eclesiastes 7:12

RESUMO

O reconhecimento de ações humanas é um problema desafiador em visão computacional que tem potenciais áreas de aplicações. Para descrever o principal movimento do vídeo um novo descritor de movimento é proposto neste trabalho. Este trabalho combina dois métodos para estimar o movimento entre as imagens: casamento de blocos e de gradiente de intensidade de brilho da imagem. Neste trabalho usa-se um algoritmo de casamento de blocos de tamanho variável para extrair vetores de deslocamento, os quais contém a informação de movimento. Estes vetores são computados em uma sequência de frames obtendo a trajetória do bloco, que possui a informação temporal. Os vetores obtidos através do casamento de blocos são usados para clusterizar as trajetórias esparsas de acordo com a forma. O método proposto computa essa informação para obter tensores de orientação e gerar o descritor final. Este descritor é chamado de autodescritor porque depende apenas do vídeo de entrada. O tensor usado como descritor global é avaliado através da classificação dos vídeos das bases de dados KTH, UCF11 e Hollywood2 com o classificador não linear SVM. Os resultados indicam que este método de trajetórias esparsas é competitivo comparado ao já conhecido método de trajetórias densas, usando tensores de orientação, além de requerer menos esforço computacional.

Palavras-chave: Casamento de blocos. Reconhecimento de ações humanas. Autodescritor. Trajetórias esparsas e densas. Clusterização de trajetórias.

ABSTRACT

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

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

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

4SS Four Step Search

ARPS Adaptive Rood Pattern Search

BoW Bag-of-words

BMA Block Matching Algorithm

FAST Features from Accelerated Segment Test

FS Full Search

GMM Gaussian mixture modeling

HOG Histogram of Oriented Gradient

ME Motion Estimation

NCC Normalized Cross-Correlation

PRA Pel-recursive Algorithm

PF Prominent Feature

RANSAC RANdom SAmple Consensus

ROI Region of interest

SAD Sum of Absolute Differences

STCK Spatial-temporal context kernel

SURF Speeded Up Robust Features

SVM Support Vector Machine

VSBMA Variable Size Block Matching Algorithm

VLAD Vector of Local Aggregated Descriptor

CONTENTS

1 INTRODUCTION

Human action recognition is a challenging problem in Computer Vision which has been an active area of research in recent decades. Works in this field are interested in examining activities in images, whether they are still images, as in (HOAI; ZISSERMAN, 2010), in which two independent images are examined, or frames captured from a video sequence, as in (YUAN et al., 2012; YI; LIN, 2013; VRIGKAS et al., 2014; TABIA et al., 2012; JAIN et al., 2013; WANG et al., 2013c; MURTHY; GOECKE, 2013).

Despite all efforts in this field, there are still difficulties such as changes in scale and viewpoint, partial occlusion, camera motion and background occlusion. Some recent works tried to solve these issues. For example, Wang et al. (2013c) used interest points to solve the problem of camera motion. Also, in (WANG et al., 2013a), the authors were able to improve human pose in such a way that it is almost unaffected by changes in scale. Nevertheless, it is difficult to solve all these problems simultaneously.

Human action recognition is basically composed of two stages: the motion information modeling and its classification. This information is extracted from movements of different parts of the human body and from their interactions. Several works have opted to use space-time representations, as in (WANG et al., 2013a; JI et al., 2010; KOBAYASHI; OTSU, 2012), because they provide better models for motion within the video, when compared to representations that solely use spatial information.

Over the recent years, many researchers have been using video descriptors in order to represent motion information for action recognition. In (MOTA et al., 2013; SAD et al., 2013; PEREZ et al., 2012; MOTA et al., 2012), orientation tensors were used in this context for describing video information. The tensor keeps data which represent the relationship between vectors coefficients associated with the original motion information. Tensors are also used in this dissertation, where the descriptor is the result of the concatenation of separate cluster vectors obtained from cross products between two different motion information.

Two different approaches are used here in order to detect motion displacement: block matching and brightness gradient between two consecutive frames. The cross product between the vectors of these two methods is calculated, and then this resulting vector

is clustered according to the mean angle of the block matching trajectory vectors. As in other works mentioned before, spatial-temporal motion information is used for better representing the motion.

In order to evaluate the video descriptor obtained by the proposed method, well-known datasets, such as KTH Schuldt et al. (2004), UCF (LIU et al., 2009) and Hollywood2 Marszalek et al. (2009), were used. Also, a Support Vector Machine (SVM) classifier was employed. Test results will be presented and discussed in Chapter 4 after thorough explanation of the method.

1.1 MOTIVATION

Human action recognition has many potential applications, such as video-surveillance systems and automatic annotation of video archives, while also being important for improving human-computer interaction.

Video surveillance systems usually have a set of cameras which are supervised by a human. However, human supervising can be flawed. Automatic supervision could bring several benefits. For instance, human-induced errors could be minimized or even eliminated and data could be processed without interruptions. This type of system would also be able to monitor children and elderlies, thus preventing domestic accidents.

We can see more applications of human action recognition in areas such as robotics for human behavior characterization (LALLÉE et al., 2010), video retrieval (LIN et al., 2012), content-based video searching (YU; YANG, 2005), entertainment (CHUNG et al., 2005), user interfaces (RAUTARAY; AGRAWAL, 2012) and sports (CHAMBERS et al., 2002).

1.2 PROBLEM DEFINITION

The main problem in this work is to define a descriptor, i.e. a vector, which describes the main action performed in it. In real videos, there are several types of movements, like camera or background motion, for instance. However, while the interest lies in the dominant human motion, our descriptor based on block matching and brightness variation actually represents the global motion in an orientation tensor.

1.3 OBJECTIVES

The main objective of this work is to present a method for human action recognition which uses a tensor-based descriptor formed by sparse trajectories. We aim to develop a method that can achieve good recognition rates and low computational effort when compared to the current state-of-the-art.

Additionally, we have the following objectives:

- Providing means for obtaining motion trajectories by using the block matching approach.
- Showing that motion trajectories yield better results than using only information of two frames.
- Demonstrating that sparse and cluster trajectories can both achieve as good results as dense trajectories approaches, such as the one presented in Caetano (2014), while requiring minor computational effort.
- Using a histogram to summarize the motion vectors.

2 RELATED WORKS

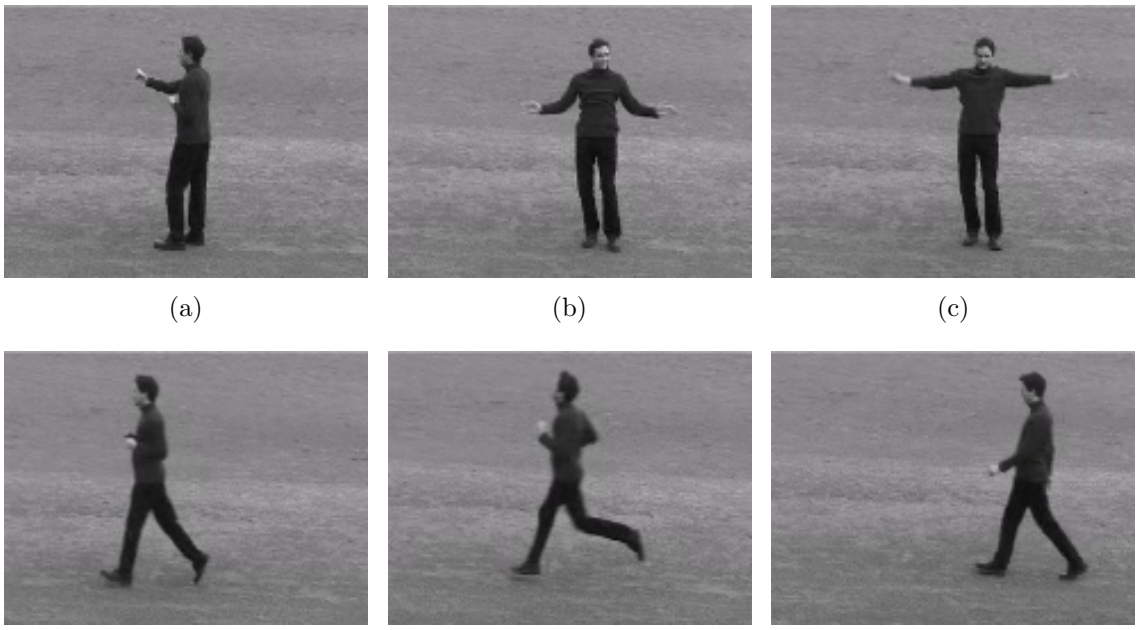
In this chapter, the main works related to this dissertation are presented, starting with datasets widely used to evaluate approaches for human recognition. These datasets are composed by many common human actions that can reproduce satisfactorily the daily actions.

Section 2.2 presents some works that use the block matching approach to better estimate motion. Section 2.3 discusses works that make use of the idea of trajectories, which served as inspiration for this work. Finally, by the end of the chapter, some other relevant works based on different approaches are discussed.

2.1 RELATED DATASETS

A common approach when working with motion estimation is to use datasets for evaluating and validating experiments. Ideally, these datasets should be widespread enough so they can serve as a comparison parameter between different works. In this work, three known datasets were used: KTH, UCF11 and Hollywood2. These datasets are extensively used by researchers throughout the literature.

The first one mentioned above, called KTH, was introduced by Schuldt et al. (2004). It is considered relatively simple, being the most used dataset for human action recognition. It is composed of six different types of human actions: boxing, clapping, waving, jogging, walking and running. These actions are performed several times by 25 actors in 4 different scenarios: outdoors, outdoors with camera zoom, outdoors with changed clothes and indoors. The KTH dataset has a total amount of 2,391 sequences with homogeneous backgrounds and a static camera. The sequences are divided into three sets: a training set with 8 people, a validation set with 8 people and a test set with 9 people. The first set is used for training a classifier, the second one is used to optimize the parameters of the classifier, and the last one is used by the classifier to predict the actions. The number of correctly predicted actions for this set is the final recognition rate. In Figure 2.1, we can see examples of this dataset, in which each video has 160×120 pixels of spatial resolution, a frame rate of 25 frames per second, and each one is about 4 seconds long. Besides proposing this database, the authors proposed a method to describe the actions.



(a) (b) (c)

(d) (e) (f)

Figure 2.1: KTH examples: (a) boxing, (b) clapping, (c) waving, (d) jogging, (e) running, (f) walking.

They used local measurements in terms of spatio-temporal interest points. The presented dataset was used in order to evaluate the method, which achieved 71.71% as recognition rate using SVM classifier.

Liu et al. (2009) presents a more complex and challenging dataset than the previous one, named the Youtube dataset (UCF11). It is composed of 11 action's categories: basketball shooting, volleyball spiking, trampoline jumping, soccer juggling, horse-back riding, cycling, diving, swinging, golf swinging, tennis swinging and walking. This dataset has some important properties to test action recognition approaches, such as different camera movements, heterogeneous backgrounds, variation in object scale, low resolution, and also several viewpoints and illumination. It has 1,168 video sequences, which are divided into 25 groups, taking into account similarities across the videos. In Figure 2.2 we can see examples of the UCF11 actions. The authors also proposed a method for action recognition in unrestricted videos. The method is based on bag-of-features (BOF) integrating both static and motion features and achieved 71.2% as recognition rate in this dataset using the Adaboost classifier.

Finally, the Hollywood2 dataset, presented by Marszalek et al. (2009), is illustrated in Figure 2.3. It is composed of 12 action's categories: answering phone, driving car,

Figure 2.2: UCF11 examples: (a) cycling, (b) trampoline jumping, (c) golf swinging, (d) soccer juggling, (e) diving, (f) horse-back riding, (g) basketball shooting, (h) volleyball spiking, (i) swinging, (j) tennis swinging, (k) walking.

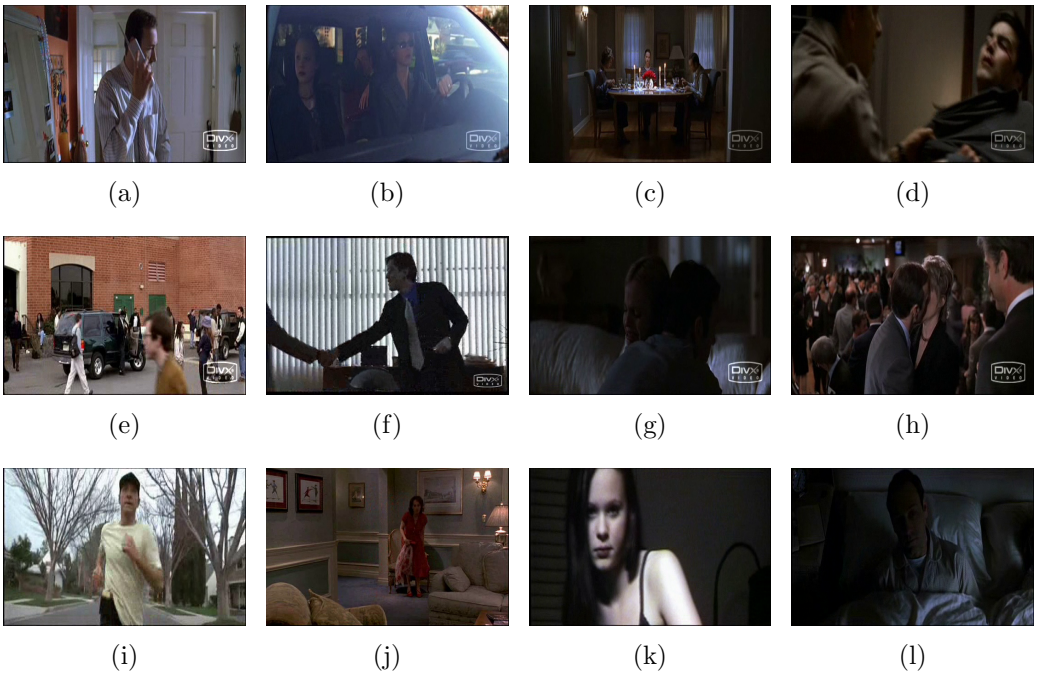


Figure 2.3: Hollywood2 examples:(a) answering phone, (b) driving car, (c) eating, (d) fighting, (e) getting out of car, (f) hand shaking, (g) hugging, (h) kissing, (i) running, (j) sitting down, (k) sitting up and (l) standing up.

eating, fighting, getting out of car, hand shaking, hugging, kissing, running, sitting down, sitting up and standing up. This dataset has 1,707 video sequences extracted from 69 different Hollywood movies. These video sequences are split in two sets: a training set with 823 videos and a test set with 884 videos. This dataset presents challenges like camera movement, different illumination conditions and heterogeneous background, which makes it of considerable value when verifying new approaches. The authors proposed visual models for scenes and actions formulated within the BOF framework, which are combined in a joint scene-action SVM-based classifier. This method achieved 37.3% as recognition rate using this database.

2.2 BLOCK MATCHING APPROACHES

The motion information obtained from block matching, which will be explained in Section 3.2, has been used in many works. Hafiane et al. (2008) presents a method for video registration based on Prominent Feature (PF) blocks. The image is divided into blocks, with a point of interest as the center, to determine the PF regions. These points of interest are extracted using structured tensors sensitive to edges and corners. Block

matching is then used to find region correspondences between two consecutive frames, using Normalized Cross-Correlation (NCC) or Sum of Absolute Differences (SAD) as a measure of similarity between blocks. Because of its normalization, NCC is less sensitive to absolute intensity changes between the reference and target images, but its computation is much more expensive than SAD's. In this work, SAD is employed as an error function and the Four Step Search (4SS) is chosen as a fast search strategy, since this is computationally more efficient than Full Search (FS), as shall be discussed in Chapter 3. A smoothing filter is also applied on each frame, in order to handle intensity outliers with less computational cost. This leads to better quality results for SAD. As in our work, it computes the direction histogram of the motion vectors but it is applied to homograph.

A block matching method is also used for extracting motion information in Amel et al. (2010). This information is used to generate a motion activity descriptor for shot boundary detection in video sequences. The method chosen for computing the motion vectors was the Adaptive Rood Pattern Search (ARPS). The authors believed that a 'low or high' intensity of motion is a good measure to determine how much a video segment is changing. Then, the identified vectors are used to calculate motion intensity, besides ranking motion according to five activity levels. Vectors with higher associated values indicate a greater probability of being a shot.

Sun et al. (2001) presents a motion activity descriptor composed of a combination of two different kinds of descriptors: a temporal one and a spatial one. The temporal descriptor is based on motion intensity, a technique similar to Amel et al. (2010) which categorize changes in scene into different intensity levels. It is obtained from the ratios between the moving blocks and the sum of all blocks in a frame. In order to be labeled as a moving block, the error must be within a margin of tolerance. Their work uses a motion intensity histogram which quantizes these ratios, whereas in our work the histogram is used to quantize vectors of motion. Their spatial descriptor is obtained through a matrix containing all the motion vectors norms from each frame.

Human action recognition has been applied not only to video sequences, but also to static images. Hoai and Zisserman (2010) proposed a method capable of recognizing human action in still images by matching rectangular blocks in order to find better correspondences between features images. It uses both the silhouette and the upper body as a model for the pose of a person. This model is used to guide the alignment between pre-

dicted human body instances for the purpose of computing registered feature descriptors. Given two images, labeled as **reference** and **probe**, it finds a bounding box of a human in the reference one and divides it into rectangular blocks. By minimizing a deformation energy between parts of the reference and the probe, a match can be established between both images.

In Figueiredo et al. (2014), we have used a block matching method to extract motion information from a video. In this method, each frame is divided into blocks of a predetermined size, and each block is matched to a correspondent block in a subsequent frame. The matching can occur in a sequence of frames in order to obtain the trajectory of a block within a frame sequence. The method calculates displacement vectors for each block and uses a histogram to quantize these vectors. This histogram is also used to calculate a tensor capable of describing a video. In the new method presented in this work, block matching is employed as in our previous method, but here a Variable Size Block Matching Algorithm (VSBMA) is used to obtain a better division of the frame.

2.3 TRAJECTORIES APPROACHES

The idea of using trajectories for extracting human activities has become increasingly popular. Yuan et al. (2012) represent these activities by a set of mid-level components and introduce a spatial-temporal context kernel (STCK). In order to extract this set of mid-level features, also called activity components, it extracts key points and track them frame by frame, forming a set of trajectories. These trajectories are assigned to clusters, depending on their appearance and motion. Then, a hierarchical descriptor is used for each activity component in order to encode its appearance, motion and shape information.

Trajectories have also been used to deal with the problem of human action recognition. Yi and Lin (2013) separately compute appearance and motion saliencies and then combine them to extract salient trajectories. This combined saliency can capture complementary information, besides being used to prune redundant trajectories, obtaining a compact and discriminative set of trajectories. Kernel histograms of both motion boundary and optical flow are applied for describing the obtained trajectories, and the bag-of-words (BoW) approach is used to classify the human actions.

Vrigkas et al. (2014) presented a method for human action recognition composed of two main steps: learning and recognition. In the first step, the optical flow and the

motion descriptor are computed, for each sequence, in a rectangular region of interest (ROI) around the human. These informations are assembled to construct motion curves, which are clustered using Gaussian mixture modeling (GMM) in order to describe each action as an unique set of these curves. During the recognition step, the optical flow curves of a probe sequence are also clustered using GMM, but they are then projected onto the training space and matched to the learned curves. Following that, the probe sequence is categorized into the learned actions.

Another work that uses the space-time trajectories idea is presented in Jain et al. (2013). The motion is decomposed into dominant and residual parts. These parts are used in the extraction of space-time trajectories and for the descriptor computation. The resulting descriptor, named DCS, is based on differential motion scalar quantities, as well as on divergence, curl and shear features (hence the acronym DCS). It uses the Vector of Local Aggregated Descriptor (VLAD) encoding technique to perform action recognition.

The residual motion discussed above also plays an important role in the method proposed by Wang et al. (2013c). In their work, two methods are combined to estimate camera motion: SURF descriptors Bay et al. (2006) and dense optical flow Wang et al. (2013b). Features points are matched and used to estimate a homograph with RANSAC (FISCHLER; BOLLES, 1981). A human detector is also employed to remove inconsistent matches originated from human activity and trajectories generated from camera motion.

Similarly to our work, Murthy and Goecke (2013) considers the importance of having a smaller but richer set of features for efficient action recognition in large scale datasets. Taking that into account, they match dense trajectories in consecutive video frames to select only a few of those trajectories. They then generate a set of ordered trajectories, employing a BoW framework equipped with an SVM classifier.

2.4 HUMAN ACTION DESCRIPTOR APPROACHES

Most approaches of human action recognition assign action labels to videos considering the whole video as a single image sequence. Unlike that, Onofri and Soda (2012) propose a method which divides the video into a number of consecutive subsequences and classifies each one of them. It combines a part-based method and the bag-of-words approach mentioned in the previous section. On the other hand, Wang et al. (2013a), as in our work, follow the usual single image sequence approach, but improve action recognition

with a method significantly different from ours. It models spatial-temporal structures of humam poses. It also incorporates additional segmentation cues and temporal constraints in the k-best human joint estimated locations obtained by the state-of-the-art method. Data mining techniques are applied in the estimated set to obtain a representation for the aforementioned spatial-temporal structures. This representation contains information about the body parts in one frame and their movements.

Another approach to human action recognition is presented by Tabia et al. (2012), which combines human detection and action recognition. As in our work, a motion histogram is used, but it is obtained in a different way. First, it locates the human figure to compute the motion histogram. Based on the optical flow, this histogram encodes the geometry of the human body and the relationships between its moving regions. After that, each motion histogram is assigned to a set of predetermined clusters, known as alphabet set. Then a bag of key-motions is constructed and classified in order to determine which class the action should be assigned to.

Other video descriptors were proposed using different methods for extracting information, such as the human action descriptor seen in Mota et al. (2013); Sad et al. (2013); Perez et al. (2012); Ji et al. (2010); Mota et al. (2012). Kläser et al. (2008) propose a local feature-based descriptor for video sequences, generalizing the Histogram of Oriented Gradient (HOG) concept to 3D. Likewise, Ji et al. (2010) extend the Features from Accelerated Segment Test (FAST) method for the 3D space. The information of shape and motion is obtained by detecting spatial and temporal features. Following Kläser et al. (2008), they produced a descriptor based on HOG3D which describes corner features. Yet another method for motion recognition that uses spatial-temporal gradients is presented by Kobayashi and Otsu (2012). It employs a motion feature extraction technique which uses auto-correlation of space-time gradients computed from three-dimensional motion shape in a video sequence. It applies the framework of bag-of-words for recognizing motion. The work presented in this dissertation produces a global descriptor which uses motion estimation information extracted in a different way from the three works described above.

Finally, Mota et al. (2013) presented a tensor motion descriptor for video sequences using optical flow and HOG3D information. In that work, they use an aggregation tensor-based technique, combining two descriptors. One of them carries polynomial coefficients

which approximate optical flow, and the other one carries data from HOGs. This descriptor is evaluated by a SVM classifier using KTH, UCF11 and Hollywood2 datasets. Caetano (2014) also uses orientation tensors as motion descriptors, besides using dense optical flow trajectories. For each optical flow point, he computes the cross product between the trajectory displacement vector and the 3D gradient vector in a window around that point. The optical flow displacement vectors are also used for clustering based on trajectories shape. According to this clustering, the resulting motion vectors are grouped and represented by a tensor.

In our work, the approach of using block matching vectors considerably reduces the effort needed for tracking motion, as compared to the use of HOG3D and dense trajectories. In the next chapter, some important fundamentals will be presented for a better understanding of the method proposed in this work.

3 FUNDAMENTALS

In this chapter the main fundamentals of this work are briefly presented. First, we present the concepts and two mainstream techniques of Motion Estimation are introduced, followed by an explanation about Block Matching, the motion estimation technique used in this work. In the Block Matching section, the search strategy and the error function chosen to match the blocks are explained. Next, the idea of using trajectories as a mean for enhancing the block matching method is discussed. Finally, a way of quantizing three-dimensional motion vectors into histogram is presented, followed by the introduction of the k-means clustering algorithms. All of these concepts are essential to understanding the method proposed in this dissertation.

3.1 MOTION ESTIMATION

Motion estimation (ME) is the process of determining motion vectors which describe the transformation from a 2D image to another. The basic premise of motion estimation is that, in most cases, successive video frames may contain the same objects, whether they are still or moving. The movement of objects in an image sequence is examined in an attempt to obtain the best vectors representing the estimated motion. These motion vectors can approximate the motion of a real video camera, such as rotation and translation, through a translational model representation. They may represent the whole image or just specific parts of it, such as rectangular blocks. The first case is known as global motion estimation.

There are two mainstream techniques of motion estimation: pel-recursive algorithm (PRA) (NETRAVALI; ROBBINS, 1979) and block-matching algorithm (BMA) (JAIN; JAIN, 1981). PRAs are based on iterative refining of motion estimation for individual pels by gradient methods. BMAs, on the other hand, estimate motion of rectangular blocks and produce one motion vector for each block, under the assumption that all the pels within a block have the same motion activity. In general, BMAs are more suitable for a simple hardware realization because of their regularity and simplicity.

Motion estimation became a powerful technique to eliminate temporal redundancy due to high correlation between consecutive frames. In real video scenes, motion is difficult to

estimate and may require large amounts of processing such as combination of translation and rotation. However, translational motion is easily estimated.

In this work, motion estimation is used to find the best vector which represents the subimage displacement between two consecutive frames. The next section will present the block matching method used here to estimate this vector.

3.2 BLOCK MATCHING

A Block Matching Algorithm computes matching blocks between two digital video frames for the purposes of motion estimation. This technique relies on the assumption that pixel patterns, which may correspond to objects or background, move within the frame sequence in characteristic ways. These characteristics help on matching corresponding objects on the subsequent frame. A block matching algorithm divides an image into rectangular blocks to obtain vectors with more accurate information.

A rectangular $s_x \times s_y$ block of an image with domain D is defined as a set of pixels $B = \{(x, y) \in D | b_x \leq x \leq b_x + s_x, b_y \leq y \leq b_y + s_y\}$, where b_x and b_y are the coordinates for the top-left pixel of the block. These coordinates are used to identify the whole block.

For each block in a frame (the 'reference frame'), the algorithm searches for a corresponding block in the next frame (the 'target frame'). The set of examined pixels on the target frame define the search window. In order to reduce the computational cost associated with the analysis of a whole frame, a restriction is imposed to the search window. This search window is centered on the block in the reference frame which is being matched, using a chosen search strategy, while also minimizing an error function. These details will be discussed in the following two sections.

3.2.1 SEARCH STRATEGY

The search strategy is an important part of the block matching algorithm. It is responsible for exploring the search window in order to find the best match for a block. The main goal of a search strategy is to reduce BMA's computational effort, retaining the quality of the results when compared to strategies which examine each block in the search window.

The simplest and generally most accurate strategy is the Full Search (FS). It attempts to match every block in the reference frame with every possible block inside the search

window on the target frame. Despite being simple and accurate, it is an expensive strategy. For example, in a 15×15 search window, it needs 255 comparisons to find the best match of a block.

Many other strategies have been proposed over the years, such as New Three Step Search (LI et al., 1994), Four Step Search (PO; MA, 1996), Diamond Search (ZHU; MA, 2000) and Adaptive Rood Pattern Search (NIE; MA, 2002). The Four Step Search (4SS) was the choice for this work. This is a steepest descent based strategy which produces results as good as the full search, while requiring much less computational resources. It is comprised of four steps applied to a search window of 15×15 pixels. During these steps, 4SS makes use of three different search patterns, which are illustrated in Figure 3.1 and described below. Differently from the FS strategy, 4SS checks, in a 15×15 window, only 17 points in the best case scenario and 27 points in the worst case scenario.

The first step starts at the center of the window and searches for matches at 9 locations in a 5×5 window. The function error, which will be explained in subsection 3.2.2, is calculated for each match of the search. If the smallest error is found in the central location, the search jumps to the fourth step. This may also happen during the second or the third step. If the smallest error is found at any one of the other eight points, the point containing the error is set as the center for the second step.

The second step tries to match blocks at 3 or 5 pixels, depending on the central pixel location. This ensures that points already visited in the previous step will not be evaluated again. Here we have two possibilities for the smallest error detected in the first step. When it is found at the side of the pattern, the second step tries to match the block using 3 of its neighbors. When it is found, however, at a corner, the block matching is attempted with 5 of its neighbors. The third step works the same way as the second step.

The fourth and final step is similar to the first, but it searches in a 3×3 window. Besides, the match with the smallest error is considered the best one for the block.

Figure 3.2 shows the 4SS's three search patterns. The first step pattern is shown in Figure 3.2(a) and represented as a blue square in Figure 3.1. Figures 3.2(b) and 3.2(c) show the patterns which can be used in the second and third steps. In Figure 3.1, these patterns are represented as black dots and green triangles, respectively. Figure 3.2(d) shows the fourth step pattern which is represented as a red diamond in Figure 3.1.

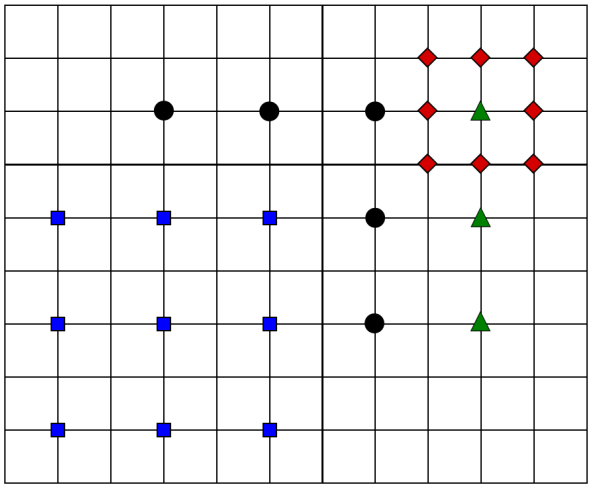
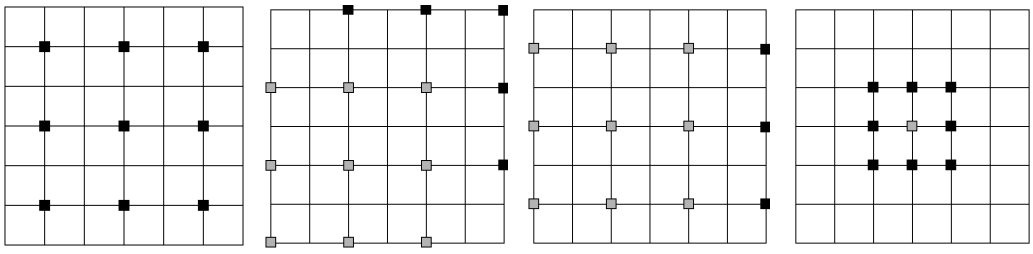


Figure 3.1: Example of 4SS steps.



(a) (b) (c) (d)

Figure 3.2: 4SS patterns: (a) first step pattern; (b-c) second and third possible step patterns; (d) fourth step pattern. The black pixels are the analyzed points and the gray ones are the previously analyzed points.

3.2.2 ERROR FUNCTION

The error function is a measure of similarity between two blocks in terms of pixel intensities. It is the single criterion to find the best match for a block in BMA. The best match is obtained by minimizing the error function.

In this work, the Sum of Absolute Differences (SAD) (PO; MA, 1996) was used as the error function between two subimages (blocks, in this case) f and g in a measurement window W. SAD accumulates the absolute differences in the measurement window W. It is defined as follows:

$$E_{SAD}(d_x, d_y) = \sum_{x,y \in W} |f(x, y) - g(x + d_x, y + d_y)|, \tag{3.1}$$

where (x, y) are the pixel coordinates and (d_x, d_y) is the motion vector.

3.3 TRAJECTORIES

A sequence of successive frames could be used in order to track object displacement generating trajectories, as in Kanade et al. (1981); Sun et al. (2009); Wang et al. (2013b); Caetano (2014).

A trajectory is a set of vectors that describe the displacement of a point into a sequence of frames. Given a point $p_i^k = (x, y)$ of index i in a frame k, the trajectory S_i^k of this point is composed by a set of vectors $\{\vec{v}_{i,0}^{\,k}, \vec{v}_{i,1}^{\,k}, ..., \vec{v}_{i,l}^{\,k}\}$, where l is the size of the trajectory. These vectors are determined using the difference between the point position in the next frame $k + 1$ and in the actual frame k, as:

$$\vec{v}_{i,j}^{\,k} = p_{i,j}^{k+1} - p_{i,j}^{k} \ , \tag{3.2}$$

where j is the index of the frame in the trajectory.

The current trend in trajectory-based methods is presented by Wang et al. (2013c) and is called 'dense trajectories'. In this kind of trajectory, the frame t is densely sampled in points, using a grid spaced by a pre-determined number of pixels in different scale spaces. Points located in areas without important motion are removed using an threshold criteria similar to Harris detector (HARRIS; STEPHENS, 1988). The remaining points are then used to represent the whole frame. Even after this filtering, a significant amount of points

still remains. For each one of these points a trajectory is associated, hence the name 'dense trajectories'. The displacement vector $\vec{v}_{i,j}^{\,k}$ in this case is the value of the optical flow in that point and this vector is used to calculate the next point of the trajectory. The set of all these optical flow vectors form the trajectory of each initial point.

In our work, we propose the use of sparse trajectories, as an alternative to the high computational effort of dense trajectories. Sparse trajectories are trajectories of points sparsely sampled. In our case, these points are sampled using a Variable Size Block Matching Algorithm (VSBMA), as will be explained in chapter 4.

3.4 CROSS PRODUCT OF TRAJECTORIES AND BRIGHTNESS GRADIENTS

Caetano (2014) proposes to add a temporal coordinate in the motion vector $\vec{v}_{i,j}^{\,k} = (x, y, z)$, making $z = 1$, meaning a displacement of one frame. This work proposed a combination of dense trajectory and brightness gradient vectors calculated in (x, y, t) around each point of the trajectory.

The proposed combination yields a vector set $C_{i,j}^{k}$, which is calculated using the cross product between each brightness gradient vector $\vec{g}_q^{\,k}$ in a window $M_{p_{i,j}^{\,k}}$ around the point $p_{i,j}^{k}$ and the trajectory displacement vector of that point. This vector set is generated by using:

$$C_{i,j}^{k} = \{\vec{c}_q^{\,k} \mid q \in M_{p_{i,j}^{\,k}}, \vec{c}_q^{\,k} = \vec{v}_{i,j}^{\,k} \times \vec{g}_q^{\,k}\} \ . \tag{3.3}$$

An interesting property of cross products is the fact that the resulting vector is perpendicular to the directions of both original vectors, so it encodes some information about those directions. Another property of interest associates the magnitude of the resulting vector with the sine of the angle between the original vectors. This information can be used to encode the tendency of the trajectory and the gradient to move together. If vectors are parallel, the resulting vector magnitude is 0. If they are perpendicular, the magnitude is equal to the product of the original magnitudes.

Caetano (2014) shows that the combination of dense trajectories and local brightness gradients using cross product improves recognition rates in comparison with approaches using only gradients. Its better results are 94.1% for KTH dataset and 46.5% for Hollywood2. Our work uses the cross product method for this reason. However, we propose

the use of sparse trajectories which requires much less computational effort. Furthermore, our method results can be fairly compared with its results.

3.4.1 HISTOGRAM OF THREE-DIMENSIONAL VECTORS

Histogram is a statistical tool used both in Caetano (2014) as in our work to reduce the dimensionality of the vectors obtained from the cross product.

In order to generate the histogram, each vector $\vec{c}_q^{\,k} = (x, y, z)$ is converted to spherical coordinates with $r = \sqrt{x^2 + y^2 + z^2}$, $\theta = \arccos(\frac{z}{r})$ and $\phi = \arctan(\frac{y}{x})$. The angles $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$ represent the vector orientation and r represents the vector magnitude.

The histogram of $C_{i,j}^k$ is denoted as $\vec{h}_{i,j}^{\,k} = (h_{1,1}, h_{2,1}, ..., h_{b_\phi, b_\theta - 1}, h_{b_\phi, b_\theta})$, where b_θ and b_ϕ are the number of bins for the coordinates θ and ϕ, respectively. The histogram components are calculated accumulating the vector magnitude in a bin corresponding to the vector angles, as the following equation:

$$h_{l,t} = \sum r_{\vec{c}_q^{\,k}} \ , \tag{3.4}$$

where l, t are bin indexes correspond to the vector angles ϕ and θ,respectively.

3.5 K-MEANS

K-means is an unsupervised clustering algorithm first introduced by MacQueen et al. (1967). It groups samples into n_c clusters, where n_c is defined a priori. First, each cluster receives a centroid, which can be random or arbitrarily chosen. During this initialization stage, the ideal configuration is to have the centroids as far away as possible from each other. Then, each sample is assigned to the closest centroid's cluster and the positions of the centroids are recalculated as the mean coordinates of the samples within the same cluster.

These assignments and centroid updates stages are repeated until either one of the following criteria are met: no more changes are observed, a maximum number of iterations is reached, or the sum of intra-cluster distances fall beneath a predetermined value.

4 PROPOSED METHOD

A general overview of the proposed method are presented in the Section 4.1 and below, three ways to use our method is presented in the Section 4.2.

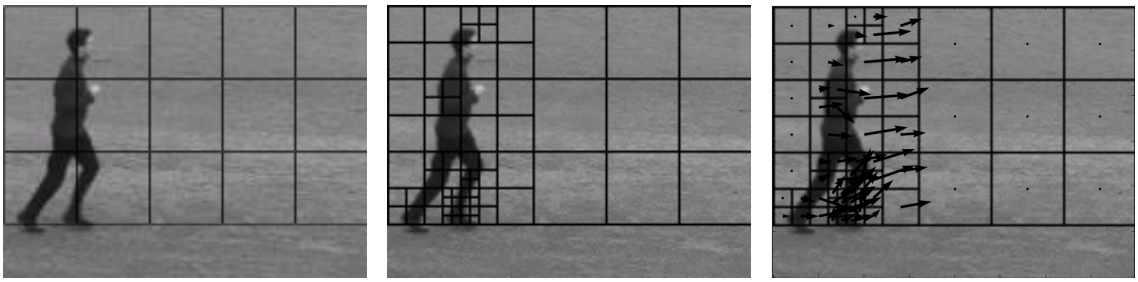
4.1 METHOD OVERVIEW

The base of the proposed method will be presented in the following sections. The block displacement is calculated using the 4SS block matching method discussed in Section 3.2.1. The resulting vector is used in the computation of the cross product with the brightness gradient in the block area. The resulting three-dimensional vectors are clustered according to the average of the angles of trajectory vectors.

4.1.1 COMPUTING SPARSE TRAJECTORIES

The input of our method is a video, i.e., a set of frames $V = \{F_k\}$, where $k \in [1, n_\mathrm{f}]$ and n_f is the number of frames. The goal is to characterize a video sequence action, so the movement is described for $k - 1$ frames, because at least one successor frame is needed for the calculation. We also need to calculate the displacement block between the first two frames of the sequence. For that purpose, a Variable Size Block Matching Algorithm (VSBMA) is used. The frame k of the video is subdivided into $n_x \times n_y$ non-overlapping blocks of exactly $s_0 \times s_0$ pixels, where s_0 is a initial block size fixed for the first frame. If the frame dimension is not a multiple of s_0, the remaining right and bottom pixels do not form blocks, as can be seen in Figure 4.1(a). The algorithm searches for each block from the reference frame in the target frame based on any search strategy of BMA. If the match error found is greater than a fixed threshold, the block is split into four half-sized blocks until the error is below the threshold or it reaches a predetermined smallest permitted size s_s. A quadtree is used for this purpose with leaf nodes corresponding to blocks of varying sizes. The objective is to make the edge of the blocks coincide with the border of objects in the scene, forming regions with uniform displacement. After this block division, a final number of blocks n_b is reached, where $n_b \geq n_x \cdot n_y$, as can be seen in Figure 4.1(b).

VSBMA is used here with a 4SS strategy to generate the displacement vectors map, as explained in Section 3.2. For each block B_i, displacement vectors $\vec{v}_i{}^k = (x, y) \in \mathbb{R}^2$ are



(a) 3x5 blocks of 32x32 pixels (b) 69 blocks of variable sizes (c) VSBMA displacement vectors map

Figure 4.1: An example of motion estimation using VSBMA

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

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

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

Figure 4.2 shows an example of the block trajectory scheme. The dark red block in frame k is matched with a light red block in frame $k+1$, generating a displacement vector. This vector (frame $k+1$) indicates the position of the new reference block (light red block in frame $k + 1$), and this block is used to make a second match between frame $k + 1$ and frame $k + 2$, generating another displacement vector. This example uses three frames to form the block trajectory, which means there are two displacement vectors.

The vector describing the displacement of a block between two consecutive frames is defined as $\vec{v}_{i,j}^{\,k} = (x, y) \in \mathbb{R}^2$, where k is the index of the initial frame of the sequence and $j \in [1, t]$ is the index of the reference frame used to compute the vector. The set of these

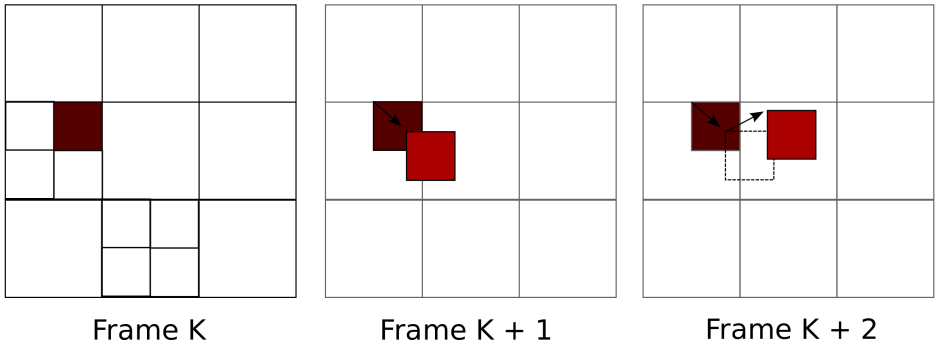


Figure 4.2: Example of the block trajectory scheme. The first match for frame k generates a displacement vector. The vector found (frame $k+1$) indicates the position of the new reference block, and this block is used to make a second match between frame $k+1$ and frame $k+2$.

t vectors of all blocks form a map of motion vectors referent to k-th frame, as can be seen in Figure 4.1(c), where $t = 4$.

4.1.2 GENERATING HISTOGRAMS USING CROSS PRODUCT VECTORS

The same reference frames used to compute the BM are used as reference frame to compute the brightness gradient. These brightness gradient vectors are used to calculate the cross product with BM displacement vectors. As these displacement vectors have only the spatial components (x, y), a third component representing the time displacement is added, in order to make possible the cross product computation. As only a single frame displacement is considered, the temporal coordinate is set to 1, as explained in Section 3.4.

We calculate a vector set which represents this cross products using

$$C_{i,j}^{k} = \{\vec{c}_q^{\,k} \mid q \in B_{i,j}^{k},\ \vec{c}_q^{\,k} = \vec{v}_{i,j}^{\,k} \times \vec{g}_q^{\,k}\}, \tag{4.1}$$

where q is a point in the block $B_{i,j}^{k}$ and $\vec{g}_q^{\,k}$ is the brightness gradient in that point. Figure 4.3 illustrates these vectors.

The motion vector set obtained from each block is represented by a motion estimation histogram. All the resulting vectors in $C_{i,j}^{k}$ are converted to equivalent spherical coordinates and quantized into histogram $\vec{h}_{i,j}^{\,k}$, as mentioned in Section 3.4. Then, we normalize

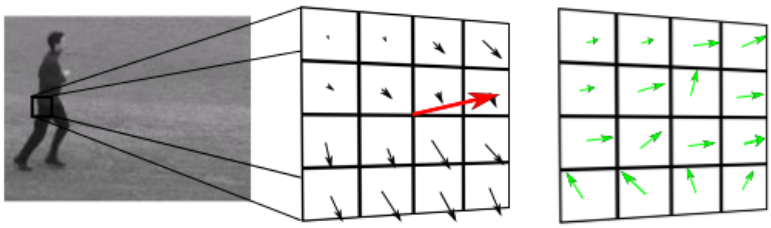


Figure 4.3: An example of cross product between the VSBMA vector (in red) and the block gradient vectors (in black).

these histograms, as is common in computer vision, to make it invariant to brightness magnitude. We have tested the L^1 and L^2 norm for histogram normalization and the first one obtain better results.

4.1.3 CLUSTERING THE TRAJECTORIES OF A FRAME

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

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

$$a^k_{i,j} = \cos^{-1}\left(\frac{\vec{v}_{i,j}{}^k \cdot \vec{v}_{i,j+1}{}^k}{\| \vec{v}_{i,j}{}^k \| \cdot \| \vec{v}_{i,j+1}{}^k \|}\right), \tag{4.2}$$

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

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

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

4.1.4 GENERATING THE FRAME DESCRIPTOR

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

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

$$\mathbf{T}_i^k = \sum_{j=0}^{t} \mathbf{T}_{i,j}^{\,k}. \tag{4.3}$$

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

$$\mathbf{T}_c^k = \sum_{i \in X_c} \mathbf{T}_i^k. \tag{4.4}$$

As proposed by Caetano (2014), the clusters c of each frame are ordered using the ascending angles ϕ and θ. The tensors $\mathbf{T}_c^k$ follow the same ordering. It ensure that the vectors with similar angles tend to be in the same cluster number for all frames.

4.1.5 GENERATING THE FINAL VIDEO DESCRIPTOR

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

$$\mathbf{T}_c = \sum_{k=0}^{n_\mathrm{f}} \mathbf{T}_c^k, \tag{4.5}$$

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

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

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

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

4.2 VARIATIONS OF THE METHOD

We presented the base of our method, but during the development of this work we tested it in three different ways. The differences are entirely concentrated in the first stage, presented in Section 4.1.1, where the sparse trajectories are computed using block matching. The method of block division and the trajectory formation can be different. The subsequent stages remain similar for all experiments.

Section 4.2.1 presents the first variation, in which the process of block division occurs between the first two frames of the sequence and no treatment of the homogeneous area is employed. The second variation, presented in Section 4.2.2, is an evolution of the first one, where the homogeneous area of the frame is treated. Finally, the third variation, presented in Section 4.2.3, modifies the block division process, by allowing it to occur in every frame of the sequence.

4.2.1 FIRST VARIATION

This first variation is a result of the steps outlined in the previous sections. First, VSBMA displacement vectors are computed using t pairs of adjacent frames, as shown in Section 4.1.1. In this phase, we use the pair of frames t_0 in the block division process, and this final division defines the size of each block. This size remains the same for the processing in the next frames. Besides, background information is not treated in this first variation of the method.

4.2.2 SECOND VARIATION

In order to improve the information we use to compute our descriptor, we modified the initial method. It is based on the assumption that the background can add noise and worsen the results. We consider areas with little or no displacement as homogeneous areas, and thus that information is discarded during the computation of the descriptor.

In this variation, some modifications are introduced in the step presented in Section 4.1.1. After computing VSBMA displacement vectors, it only uses those with an absolute

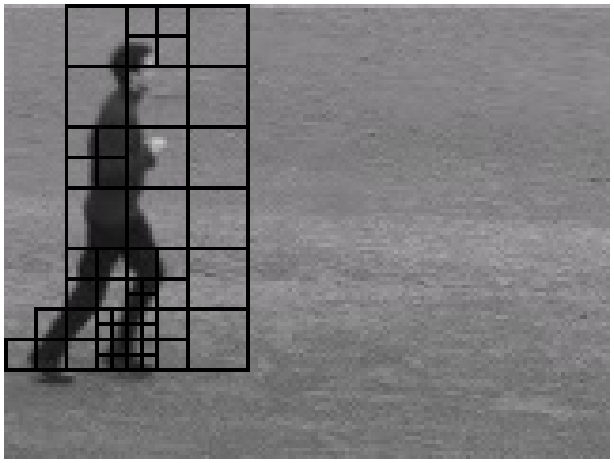


Figure 4.4: An example of a frame without the background motion information.

displacements sum greater than 2. We consider this amount is enough to remove irrelevant regions. In Figure 4.4, we can see the result of this modification. The figure shows only the blocks which contribute to our descriptor.

4.2.3 THIRD VARIATION

In the third variation, a modification is introduced in the block division part of VSBMA. We use all trajectory frames in this division. We initially divide frame f_0 into blocks of a predetermined initial size. Each block is searched in the subsequent frame f_1 and can be divided according to the match error, as explained in Section 4.1.1.

In the previous variations, we had two conditions for ending the division of a block: the match error should be smaller than the threshold or the division should result in four blocks smaller than the smallest predetermined size. In this third variation, if we reach the first condition, we try to continue the process using the next frame f_2. The division process proceeds until the block size reaches the smallest size. Also, background treatment is not present in this method variation.

In Figure 4.5, we can notice that the number of blocks increased considerably compared to the previous method. This happens because all frames contribute to the division process, and frames in the end of the sequence produce additional divisions when compared to the other variations.

In the follow chapter, we present our experiments with these three variations of our method.

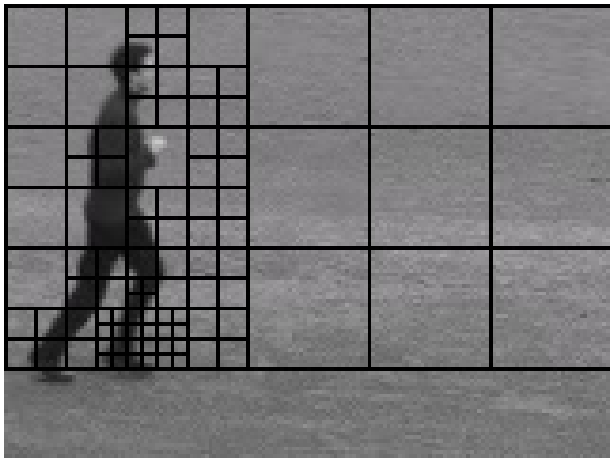


Figure 4.5: An example of a block division in the third variation

5 EXPERIMENTAL RESULTS

Besides the video, we use six parameters as input for our method. VSBMA have some important parameters, such as the initial block size, the minimum allowed size for blocks and the threshold. These parameters directly affect the final division of the blocks and, consequently, the descriptor performance.

The frame is initially divided into blocks of $s_0 \times s_0$ pixels. After this process, the blocks can have s_s as the smallest permitted block size. In the block division part of VSBMA, if the match error found is greater than a fixed threshold, the block is candidate for split. When splitting a block in four others, if the size of these new blocks are smaller than s_s, then the division does not occur.

The trajectory size is the amount of frame pairs t used in the computation of displacement vectors. This number has to be large enough to satisfactorily catch the temporal information.

The number of bins is a parameter of histogram's division. The histogram is uniformly divided into a specified number of bins. The greater this number, the lower the range of similar information and greater the tensor size generated for it. Thus, this parameter directly impacts in the size of the final descriptor.

The k-means algorithm groups samples into c clusters. As the number of clusters is increased, the final descriptor also becomes larger. This happens because each cluster has a representative tensor which is concatenated to form the video descriptor. All clustering steps are done using the stopping criterion of 500 iterations or less than 0.01% of change in the cluster membership from last iteration.

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

In order to measure the recognition rate achieved using our descriptors, we use a SVM

classifier, which takes the descriptors for the whole database. We follow the same classification protocol as (SCHULDT et al., 2004). We use both Triangle and Gaussian kernel and an one-against-all strategy for multiclass classification. We produce 6 recognition rates for each parameter combination, 3 using a triangular kernel, and 3 using a Gaussian kernel. In the following sections, we present the highest result achieved from these ones.

5.1 RESULTS FOR THE FIRST VARIATION

5.1.1 KTH EXPERIMENTS

Initially, we tested our method with three different values for the initial block size $\{(16 \times 16), (32 \times 32), (48 \times 48)\}$. The other parameters were gradually fixed based on empirical knowledge acquired during the development of this work. The smallest permitted block size was initially set to 4×4 and the threshold was set to 10. Moreover, 26 bins and 3 clusters were used. All tests were performed with {2, 4, 6, 8, 10} frames in the trajectory. In Figure 5.1, we show the recognition rates of these tests. We can note the trend of our best results in the red curve, which depicts the recognition rate for blocks with initial size equal to 32×32. We believe that this is related to having some blocks in the region without major movements when their size is 16×16, besides describing redundant information. In the best case, the method achieved a rate of 91.1%, using 6 frames as the size of the trajectory. Then we performed the next parameter tests using 32×32 initial block size.

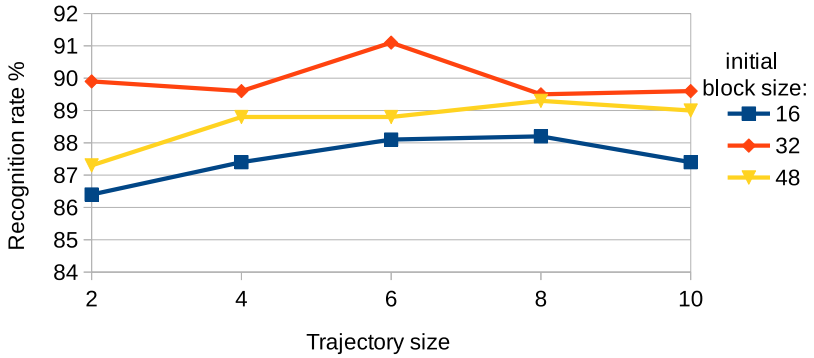


Figure 5.1: Experiments on KTH dataset with different initial block size values, 4×4 as smallest block size, 10 as threshold, 26 bins and 3 clusters.

Figure 5.2 shows the curves referent to experiments using $\{(4 \times 4), (8 \times 8)\}$ as the

smallest permitted block size. Initial block size was fixed at 32×32, threshold was set to 10, 26 bins and 3 clusters were used. One can observe that the rates for 8×8 size (shown in blue) tend to worsen as the trajectory size increases. Besides, its best result is still much below the 4×4 curve. It occurs because blocks of smaller sizes can correctly represent regions of important movement. Tests with 4×4 size also achieved 91.1% of recognition rate using 6 frames for the trajectory size, similarly to the initial block size tests. Thus, the smallest permitted block size were fixed to 4×4 for the subsequent experiments.

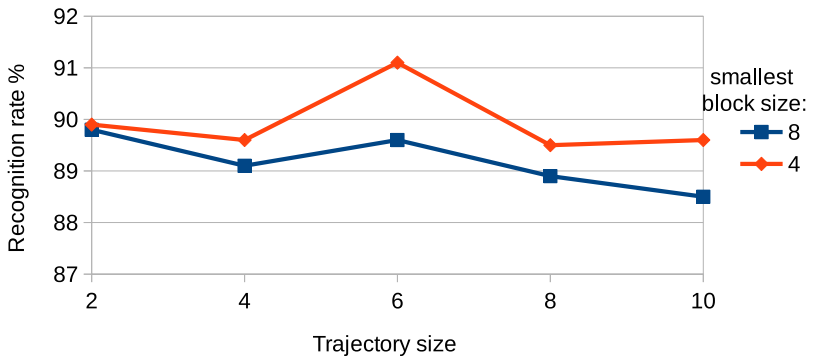


Figure 5.2: Experiments on KTH dataset with different smallest block size values, 32×32 as initial block size, 10 as threshold, 26 bins and 3 clusters.

Next, tests were performed for the following thresholds: 5, 7, 10, 15, 20, 25 and 30. Figure 5.3 shows recognition performance for each one of them. We used 32×32 as the initial block size, 4×4 as the smallest permitted block size, 26 bins and 3 clusters. We can note the threshold of 5 has the worse curve of the graphic. This value causes the formation of many trajectories and these trajectories can have noisy information. When we increase the threshold, the recognition rates improve. The same 91.1% of recognition rate was obtained for two threshold values: 10 and 25. In both cases, trajectory size was equal to 6 frames. The threshold 10 cause the generation of many trajectories and consenquently, worsening performance computing. However, considering a larger span of trajectory sizes, a threshold of 10 could maintain a better rate than when using 25. Then, we use 10 as threshold in next tests.

In Figure 5.4, we show the curves of experiments conducted for 12, 24, 26 and 30 bins. Based on previous results, we used 32×32 as the initial block size, 4×4 as the smallest

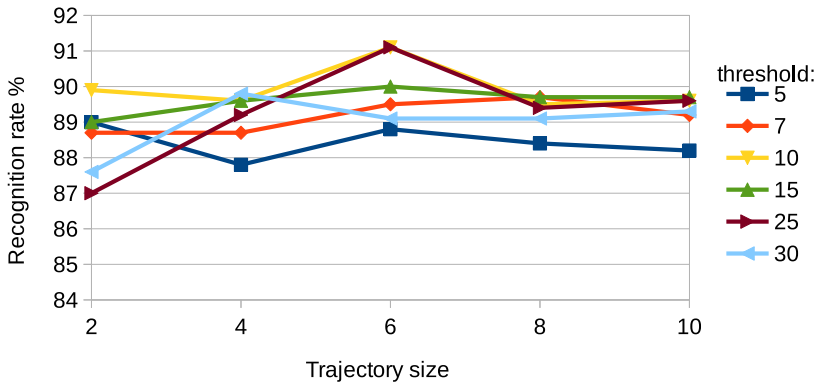


Figure 5.3: Experiments on KTH dataset with different threshold values, 32×32 as initial block size, 4×4 as smallest block size, 26 bins and 3 clusters.

permitted block size, 3 clusters and a threshold of 10 to perform these tests. The 30 and 26 bins tests achieved good recognition rates. Although experiments with 30 bins achieved better results than those with 26 bins for trajectories of size 4, 8 and 10, we decided to use 26 bins in the subsequent tests, mainly because its result was better when 6 frames were set as the trajectory size.

Small values of bins implies to large ranges of angle and, consequently, the grouping of vectors with different information. On the other hand, big values can generate empty bins, increasing the descriptor size with no information gain.

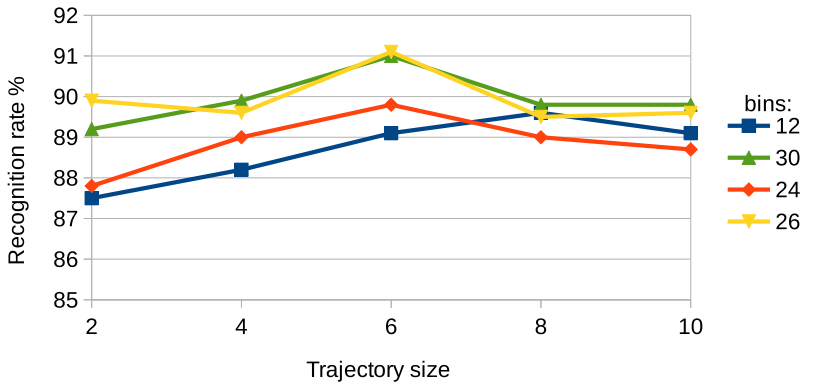


Figure 5.4: Experiments on KTH dataset with different bins values, 32×32 as initial block size, 4×4 as smallest block size, 10 as threshold and 3 clusters.

Finally, we performed tests using {1, 2, 3, 4} clusters for the k-means algorithm. All

other parameters were set to the best values previously found. We use 1 cluster in order to show that no clustering of the trajectories results in worse recogniton rate than using clustering. The best recognition rate was achieved with 3 clusters and a trajectory size of 6, as can be seen in Figure 5.5. If we observe the results for the trajectory size equal to 6, we can see that, as the number of clusters is incremented, the recognition rate only increases until a certain point, where it begins to fall. This shows a tendency for the best number of clusters for this dataset to be 3. In contrast to (CAETANO, 2014), which uses 5 clusters, as our method uses sparse trajectories, we achieve better results using a small number of clusters.

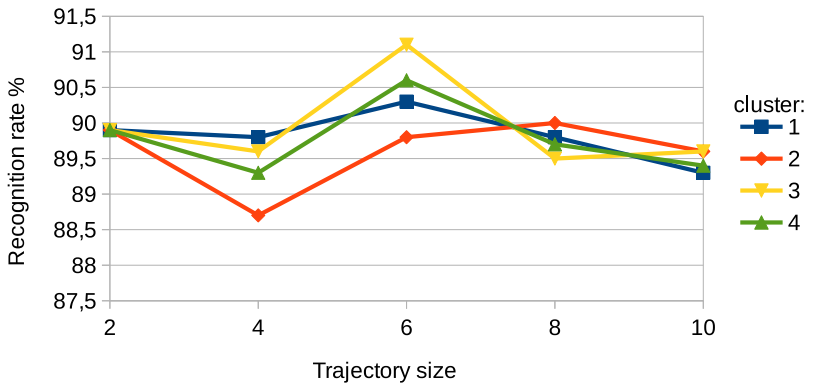


Figure 5.5: Experiments on KTH dataset with different cluster values, 32×32 as initial block size, 4×4 as smallest block size, 10 as threshold and 26 bins.

In all experiments presented, we can observe a similar behavior in which the results increase to a peak and then decrease. This peak indicates the parameter values that are likely to yield the best results in KTH dataset. In the majority of the tests, the best results were achieved with a trajectory size of 6. We tried to find a combination of parameters that would provide good results with a satisfactory number of trajectories. Ideally, this number has to be small enough so as to not require too much computing and large enough to accommodate important information.

In Table 5.1, a confusion matrix shows the best result for KTH dataset with our method, where the average recognition rate is 91.1%. The main diagonal of this matrix represents the percentage of correctly classified videos. The *walking* class has many blocks moving in the same direction, so the method is capable to obtain 100% of recognition for this motion class. This is the same reason why it has difficulty to differ *clapping* from

Table 5.1: Confusion matrix of the best result on KTH dataset with the first variation. The average recognition rate is 91.1%.

	Box	HClap	HWav	Jog	Run	Walk
Box	99.3	0.7	0.0	0.0	0.0	0.0
HClap	6.9	91.0	2.1	0.0	0.0	0.0
HWav	1.4	2.8	93.1	0.0	2.8	0.0
Jog	0.0	0.7	0.7	87.5	4.2	6.9
Run	0.0	0.0	0.0	22.9	75.7	1.4
Walk	0.0	0.0	0.0	0.0	0.0	100

boxing, where the key motion occurs in small regions of the frame. One can also notice the classical problem of differing *running* from *jogging*. The difficulties here arise from their similarity and are common to all methods in the literature.

5.1.2 UCF11 EXPERIMENTS

As the previous dataset is quite simple, we tested our descriptor in UCF11 dataset, which have more action classes and colorful videos. We performed a test using the best parameters previously found, which means, in VSBMA, 32×32 as initial block size, 4×4 as smallest block size and 10 as threshold. We use 26 bins for the histogram, 3 clusters in k-means algorithm and 6 as the trajectory size.

We achieved 65.8% as recognition rate, as can be observed in Table 5.2. Some actions have recognition rates above 80%, such as *jumping, diving, riding*. We believe this is related to the fact that these are simpler actions. On the other hand, some actions were mistaken for others, as *juggling* for *jumping*, *shooting* for *tennis*, and *walking* for *riding*. This kind of confusion occurs because theses actions are quite similar and our descriptor could not appropriately differentiate them.

The state-of-art in this dataset, presented by (WANG et al., 2013a), achieve a recognition rate of 89.9%. Despite our results are below this value, it is important to emphasize that this method has very high computational cost and is not restricted to the self-descriptor constraint.

5.1.3 HOLLYWOOD2 EXPERIMENTS

In order to test our descriptor in more real videos, we used the Hollywood2 dataset. We performed the test with the same parameters used for the UCF11 dataset, which were

Table 5.2: Confusion matrix of the best result on UCF11 dataset with first variation. The average recognition rate is 65.8%.

	Bike	Dive	Golf	Juggle	Jump	Ride	Shoot	Spike	Swing	Tennis	WDog
Bike	71.8	0.0	1.0	1.0	1.0	8.3	1.0	1.4	6.4	2.0	6.0
Dive	1.0	88.9	0.0	0.0	0.6	0.7	5.5	0.0	2.2	0.0	1.1
Golf	0.0	2.0	76.5	8.2	0.0	0.0	3.7	0.0	2.6	7.1	0.0
Juggle	1.9	0.6	3.8	50.5	12.7	1.6	4.9	2.9	6.1	8.3	6.7
Jump	1.0	1.0	1.0	4.0	80.7	0.0	1.0	0.0	4.3	2.0	5.0
Ride	4.9	1.1	0.0	0.5	0.0	84.3	0.0	1.2	1.2	0.7	6.2
Shoot	5.1	6.4	8.6	8.3	3.0	3.6	32.4	11.5	1.7	16.6	2.8
Spike	1.0	0.7	1.0	1.1	0.7	2.0	9.3	76.2	4.0	0.0	4.0
Swing	12.5	0.8	1.4	3.1	9.1	0.7	2.0	0.0	65.2	0.0	2.5
Tennis	5.3	3.2	10.5	10.1	1.1	3.2	15.9	4.1	1.1	42.5	2.9
WDog	4.8	2.5	3.3	5.1	1.8	17.5	1.8	1.6	3.7	3.3	54.7

the best parameter configuration found for the KTH dataset. As the Hollywood2 videos requires more computational resources, varying the parameters would not be viable. In 5.3 we show the average precision for each class of Hollywood2 dataset and its average recognition rate is 41.5%. Actions with the main movement in small regions, as answer phone and sit up, present the worst results of the dataset. Our result is lower than the state-of-art, preseted by (WANG et al., 2013c), that achieves 64.3% as recognition rates, but our method has low computational cost if compared with this method.

Table 5.3: Average precision for each class of Hollywood2 dataset with first variation. The average recognition rate is 41.5%.

Action	APhone	DCar	Eat	FPerson	GetOutCar	HShake
AP(%)	14.3	81.7	55.7	50.3	33.4	19.8
Action	HPerson	Kiss	Run	SDown	SUp	StandUp
AP(%)	22.0	55.3	59.6	51.3	11.2	43.7

5.2 RESULTS FOR THE SECOND VARIATION

5.2.1 KTH EXPERIMENTS

The results of our tests with second variation are summarized in Figure 5.6. We began the experimentation by fixing values for the parameters according to the best result obtained with the first variation, i.e., 32×32 as the initial block size, a block with smallest size of 4×4, a threshold of 10 for VSBMA, 26 bins in the histogram, 3 clusters for the K-means algorithm and trajectory sizes of 2, 4, 6, 8 and 10 frames.

Similarly to the approach adopted with the first variation, we tried to obtain the best values by subsequently varying each parameters as follows:

1. Initial block size values, (Fig. 5.6(a)): $\{(16 \times 16), (32 \times 32), (48 \times 48)\}$.
2. Smallest block size values, (Fig. 5.6(b)): (4×4) and (8×8).
3. Threshold values, (Fig. 5.6(c)): 5, 10, 20 and 25.
4. Bins values, (Fig. 5.6(d)): 24, 26 and 30.
5. Clusters values, (Fig. 5.6(e)): 1, 2, 3 and 4.

In Figure 5.6(a), we can see the 32×32 initial block size curve as producing the best results, with 90.7% of recognition rate when using a trajectory size equal to 2. Figure 5.6(b) shows results of experiments varying the smallest permitted block size. The (4×4) test achieved 90.7% of recognition rate as its best result, using a trajectory of size 2, while the (8×8) one only achieved 89.7% at its best. This is because smaller blocks in regions containing more movement can generate important information.

Following the analysis of the tests, the best recognition rate achieved in threshold tests was 90.7% using 10 as threshold and 2 as the trajectory size, as shown in Figure 5.6(c). This parameter is the main responsible for the division of the blocks. Thus, we believe the value found is responsible for balancing the number of blocks. It generates a few trajectories in regions with reduced movement, and a considerable amount of trajectories in regions with more movement in the image.

In Figure 5.6(d), we can see that the curve of 26 bins achieves its best recognition rate, 90.7%, using 2 as the trajectory size. Finally, we can observe in Figure 5.6(e) that the best result of clusters tests was achieved using 3 clusters for the k-means algorithm, using 2 as the trajectory size. It is important to remember that the bigger the number of bins or clusters, the greater the size of the descriptor.

This method variation presented an unexpected behaviour. In all experiments, we can notice the recognition rate fall as the trajectory size increases, unlike the results obtained for the first variation, shown in Figure 5.2. In all set of tests using the second variation the best recognition rate was achieved with 2 frames in trajectory size. So, we concluded that increase the trajectory size worsen the results.

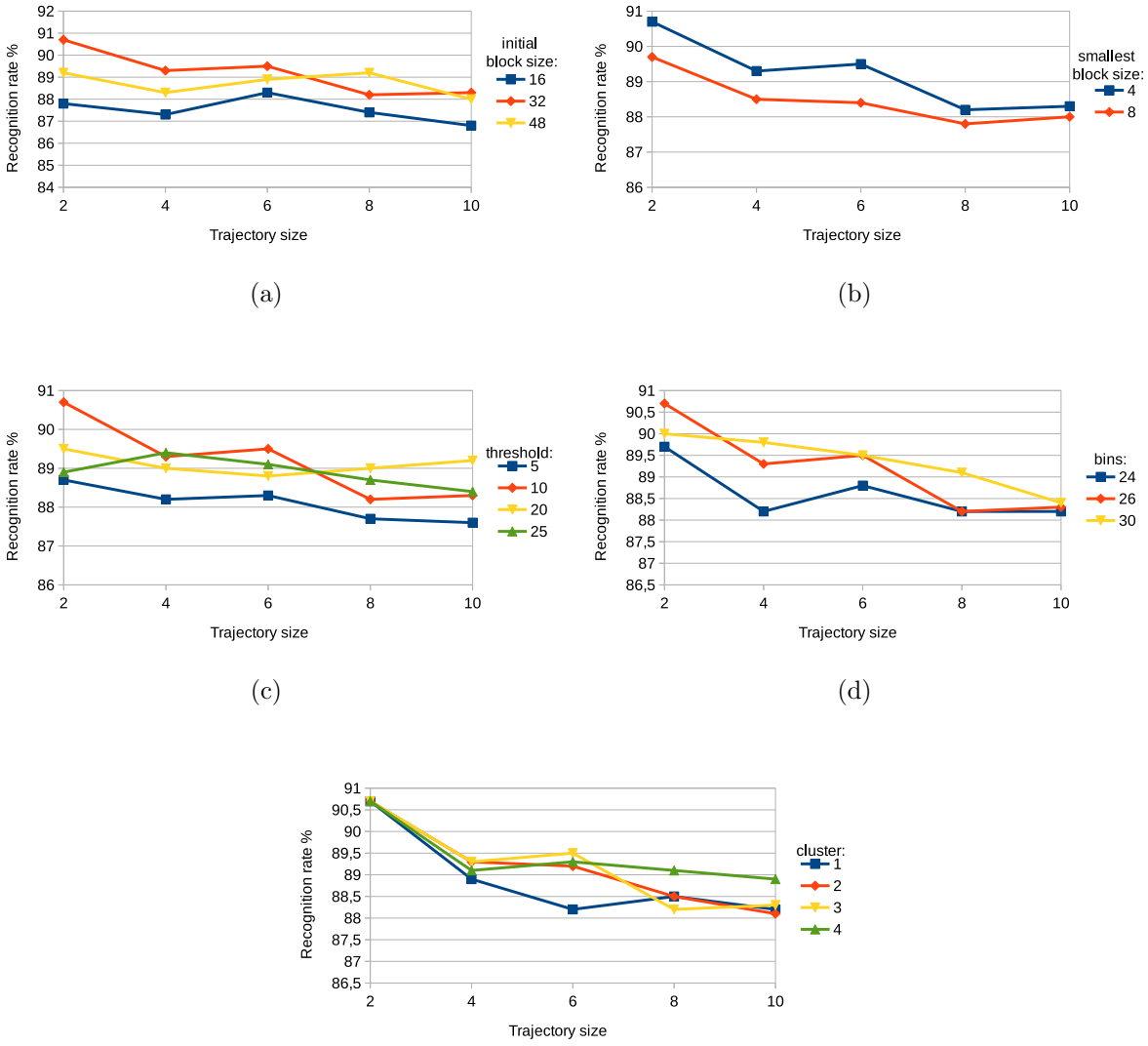


Figure 5.6: Experiments on KTH dataset with second variation

Table 5.4 presents a confusion matrix for the best result on KTH dataset, where the average recognition rate was of 90.7%. If we compare this matrix with the one associated with the first variation, presented in Section 5.1, we can see that recognition rates decrease in most action categories. Even though this experiment did not produce the expected results, investigating new approaches for background treatment may yield interesting results in the future.

Table 5.4: Confusion matrix of the best result on KTH dataset with second variation. The average recognition rate is 90.7%.

	Box	HClap	HWav	Jog	Run	Walk
Box	97.9	2.1	0.0	0.0	0.0	0.0
HClap	13.9	84.7	1.4	0.0	0.0	0.0
HWav	0.0	3.5	95.8	0.0	0.7	0.0
Jog	0.0	0.0	0.0	90.3	1.4	8.3
Run	0.0	0.0	0.0	23.6	75.7	0.7
Walk	0.0	0.0	0.0	0.0	0.0	100

5.2.2 UCF11 EXPERIMENTS

Experiments were also performed on the UCF11 dataset using the second variation. We used the parameters of the best result obtained with KTH testing. We achieved 64.7% of recognition rate, as we can see in Table 5.5. Recognition falls slightly when compared to the results of the first variation, which was 65.8%.

Table 5.5: Confusion matrix of the best result on UCF11 dataset with second variation. The average recognition rate is 64.7%.

	Bike	Dive	Golf	Juggle	Jump	Ride	Shoot	Spike	Swing	Tennis	WDog
Bike	66.1	0.0	1.0	0.0	0.5	17.3	0.0	0.0	3.8	1.7	9.6
Dive	0.0	94.1	0.0	0.00	1.2	1.2	0.5	1.2	0.0	0.0	1.7
Golf	0.0	2.8	76.3	6.4	0.0	0.0	3.1	2.6	0.0	6.1	2.6
Juggle	1.4	4.2	6.0	46.5	14.7	0.6		4.8 1.1	10.8	8.1	1.8
Jump	1.6	2.0	0.0	5.8	81.5	0.0	0.0	0.0	8.1	0.0	1.0
Ride	6.4	1.4	0.0	0.0	0.0	81.3	0.0	3.7	0.7	0.0	6.4
Shoot	6.6	10.2	5.3	8.9	2.9	3.5	36.2	11.9	5.3	6.4	2.8
Spike	2.0	3.0	1.2	2.6	1.0	4.6	3.8	68.5	2.0	7.3	4.0
Swing	10.4	0.0	0.8	3.5	14.0	0.0	2.6	0.0	64.7	0.7	3.3
Tennis	3.2	4.6	17.0	3.5	1.2	1.8	7.6	5.2	1.1	53.4	1.2
WDog	11.4	2.3	5.8	4.0	2.4	22.5	1.6	0.0	5.9	1.5	42.7

5.2.3 HOLLYWOOD2 EXPERIMENTS

Finally, we then performed tests with Hollywood2 dataset using once again the parameters of the best result found for KTH. In Table 5.6, we show the recognition rates of each class. This experiment achieved an average recognition rate of 36.1%, which is much worse than our results in Hollywood2 with the first variation, which was 41.5%. As with previous datasets, modifications introduced to this variation did not present the expected results.

Table 5.6: Average precision for each class of Hollywood2 dataset with second variation. The average recognition rate is 36.1%.

Action	APhone	DCar	Eat	FPerson	GetOutCar	HShake
AP(%)	11.5	77.1	33.9	47.3	30.7	18.7
Action	HPerson	Kiss	Run	SDown	SUp	StandUp
AP(%)	21.8	47.9	50.6	45.0	8.2	40.6

5.3 RESULTS FOR THE THIRD VARIATION

5.3.1 KTH EXPERIMENTS

The experimental results of third variation using KTH are shown in Figure 5.7. As initial parameter values, we used 32×32 as the initial block size, smallest block size of 4×4, 26 histograms bins, threshold of 10 for VSBMA, 3 clusters in K-means algorithm and 2, 4, 6, 8 and 10 frames for the trajectory size. Parameter values were varied as follows:

1. Initial block size values, Fig. 5.7(a): $\{(16 \times 16), (32 \times 32), (48 \times 48)\}$.
2. Smallest block size values, Fig. 5.7(b): (4×4) and (8×8).
3. Threshold values, Fig. 5.7(c): 5, 10, 20, 25 and 30.
4. Bins values, Fig. 5.7(d): 24, 26 and 30.
5. Clusters values, Fig. 5.7(e): 1, 2, 3, 4 and 5.

We initiated these tests by varying the initial block size. Looking at the graph of Figure 5.7(a), the curve which achieved the best result was the (32×32) one, where we obtained 89.9% of recognition rate, using 2 as the trajectory size. Note that, increasing the trajectory size does not improve results when using this parameter configuration. In Figure 5.7(b), good results are shown for a smallest block size of (4×4). Its curve shows better results than (8×8) for all trajectory sizes. This behavior has already been observed in previous tests, shown in Figures 5.2 and 5.6(b).

In the threshold tests, depicted in Figure 5.7(c), we achieved our best results when using 20 and 25 as threshold, yielding 91.1% and 91.4% of recognition rate for trajectory sizes of 4 and 6, respectively. As the 25 curve exhibits a better behavior and using this value we decrease the computational cost, the threshold 25 is used in the subsequent tests.

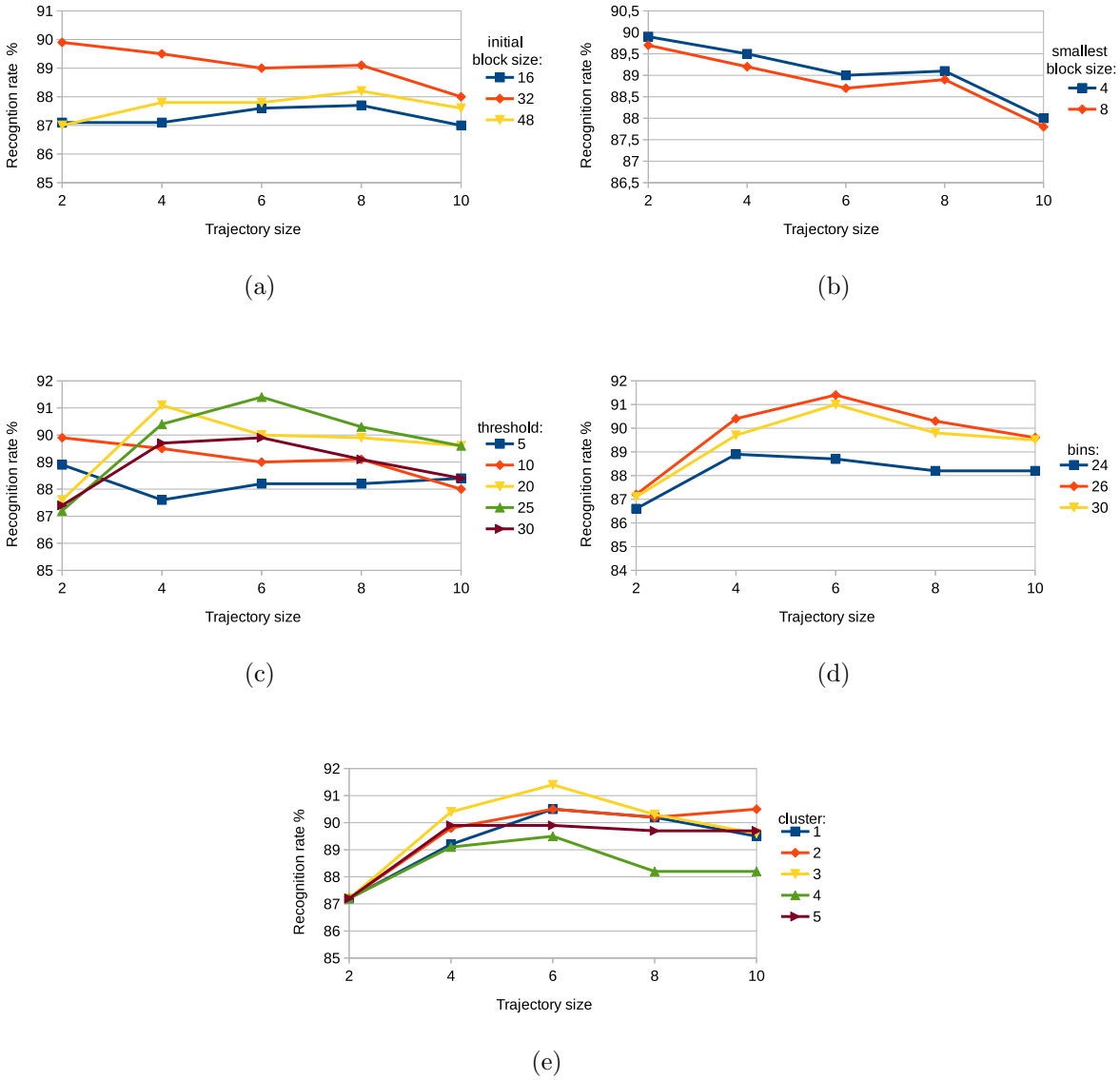


Figure 5.7: Experiments on KTH dataset with third variation

The threshold value directly influences the number of trajectories: the higher the value less trajectories are formed. As this variation form more trajectories than the previous ones, the treshold 25 maintain a reasonable amount of trajectories even being a high value.

In Figure 5.7(d), the curve for 26 bins presents the best results for all trajectory sizes tested. It achieved 91.4% of recognition rate using 6 as trajectory size. This was also the best rate obtained for clusters tests, shown in Figure 5.7(e). These set of tests has similar behavior to first variation tests, shown in Figure 5.4 and 5.5. Once again, we observed that 6 is an appropriate value for the trajectory size in this dataset.

In Table 5.7, we show the confusion matrix of our best result in KTH. In this test, we achieved a recognition rate of 91.4% using 32×32 as initial block size, 4×4 as smallest block size, 25 as threshold, 26 bins, 3 clusters and 6 as trajectory size. In *boxing*, *clapping*, *waving* and *running* classes, we achieved better recognition than using our previous method variations, shown in Table 5.1 and 5.4. Our video descriptor recognized the *boxing* action in 100% of boxing-videos. However, we had a small worsening in recognition for *jogging* and *walking* classes, comparing with first variation results. As said before, these are the two more complicated actions of this dataset.

Table 5.7: Confusion matrix of the best result on KTH dataset with third variation. The average recognition rate is 91.4%.

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

5.3.2 UCF11 TESTS

For the UCF11 dataset, we achieved 65.6% of recognition rate using the best parameters previously found. The recognition rates for each class of this test are presented in the confusion matrix represented in Table 5.8.

We can note that the obtained results are better than those found by the second variation, as it happened with the KTH tests. However, compared to the first variation, this one performs slightly worse. It can be observed in the action *shooting*, which is confused with *tennis* and *spike*. It occurs because these three action have similar arm movement as the main one of the video.

5.3.3 HOLLYWOOD2 TESTS

Concluding our experiments, we tested our best parameter configuration for this third variation against Hollywood2 dataset. In Table 5.9, we show the confusion matrix for this experiment, where we were able to achieve a recognition rate of 40.5%.

Table 5.8: Confusion matrix of the best result on UCF11 dataset with third variation. The average recognition rate is 65.6%.

	Bike	Dive	Golf	Juggle	Jump	Ride	Shoot	Spike	Swing	Tennis	WDog
Bike	70.8	0.0	0.0	1.0	0.7	8.0	2.0	1.0	9.2	1.8	5.5
Dive	2.1	87.8	0.6	0.0	1.2	0.7	6.5	0.0	0.0	0.0	1.1
Golf	0.0	2.0	75.8	5.5	0.0	0.0	3.0	1.6	1.6	10.5	0.0
Juggle	1.1	0.6	3.6	54.9	13.0	1.6		4.3 4.0	7.3	6.7	3.1
Jump	0.0	2.0	1.0	5.7	75.0	0.0	1.0	0.0	9.3	2.0	4.0
Ride	5.5	1.1	0.0	0.5	0.0	82.8	0.0	1.0	1.7	0.0	7.3
Shoot	4.4	6.7	6.7	5.8	1.7	4.4	38.0	10.0	3.0	17.2	2.2
Spike	2.0	1.7	0.0	0.0	0.7	3.0	8.3	74.2	3.8	3.3	3.0
Swing	13.3	0.8	1.4	4.6	6.5	1.2	1.0	0.5	66.9	0.0	3.7
Tennis	4.3	0.6	14.2	6.4	0.0	2.6	17.7	4.9	1.7	41.2	6.5
WDog	7.0	2.3	4.9	4.1	1.6	20.3	0.8	1.6	1.5	1.7	54.4

Once again, this variation obtained results which were better than those of the second variation, but worse than the ones obtained by the first variation. This dataset has more complex videos than KTH. Hollywood2 videos usually present more than one person. The movement of these secondary persons cause wrong matchings, with bad impact in long trajectories.

Table 5.9: Average precision for each class of Hollywood2 dataset with the third variation. The average recognition rate is 40.5%.

Action	APhone	DCar	Eat	FPerson	GetOutCar	HShake
AP(%)	13.1	80.7	50.8	45.2	32.7	20.7
Action	HPerson	Kiss	Run	SDown	SUp	StandUp
AP(%)	21.7	55.2	58.8	50.5	11.5	46.0

5.4 COMPARISON WITH STATE-OF-THE-ART

In Table 5.10, we show a comparison of our best result with state-of-the-art methods. Although it is clear that our results can not reach the state-of-the-art, that was not our initial goal. Our focus was on developing a method capable of obtaining reasonable results using sparse trajectories, which are computationally less expensive than dense trajectories.

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

Table 5.10: Comparison with state-of-the-art for KTH, UCF11 and Hollywood2 datasets.

	KTH	**UCF11**	**Hollywood2**
Klaser et al. (2008)	91.0		24.7
Perez at al. (2012)	92.0		34
Mota et al. (2013)	93.2	72.7	40.3
Wang et al. (2013)	**95.3**	**89.9**	59.9
Wang and Schmid (2013)			**64.3**
Jain at al. (2013)			62.5
Figueiredo et al. (2014)	87.7	59.5	34.9
Caetano (2014)	94.1		46.5
Our method	**91.4**	**65.8**	**41.5**

methods.

It is important to emphasize that most of these methods shown in Table 5.10 are not restricted to the self-descriptor constraint. Their video final descriptor are calculated using information of all videos of the dataset. Our descriptor, on the other hand, does not have any dependency on other videos. So, comparing our results with other self-descriptor methods, our results are better than those of Mota et al. (2013) and Perez et al. (2012) in the Hollywood2 dataset, which is, as mentioned before, the most complex set of videos used for testing in several works.

Some state-of-the-art methods shown in Table 5.10 present rates of 92% and above in KTH dataset, while our have shown a rate of 91.4%. Their results are very close to the limit of this dataset. When the recognition rate in a database reaches that point, it is very difficult to improve it further even by some tenths. Comparing our results against our first self-descriptor method, presented in (FIGUEIREDO et al., 2014), this method yields better results.

We perform some tests in order to estimate the computational effort of our method. We use a machine with Intel Xeon E5-4607, 2.20GHz, 32 GB of RAM using only 1 thread. We select 10 videos per dataset to perform this test quickly. The best parameter configuration previously found was used and our descriptors were computed with an average of 8.48, 1.66, 0.72 frames per second for the KTH, UCF11 and Hollywood2 videos, respectively. Tests in the same conditions were performed using the (CAETANO, 2014) method, which were computed with an average of 1.57, 0.61 frames per second for the KTH and Hollywood2 videos. This work does not provide results for the UCF11 database. As our method had higher processing rate, we can assume that our method requires lower computational cost than the dense trajectory method presented in (CAETANO, 2014).

6 CONCLUSION

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

We presented three variants of our method in this work and we achieved our best result with the last one. The first one uses only the two first frames of a sequence to perform the block division. After this division, the size of each block remains the same for the other frames of a sequence. This first variation achieved 91.1% of recognition rate. In order to improve our method, we presented a second variation which treats the homogeneous area. Our best result using this variation was 90.7% of recognition rate. Our third variation does not treat homogeneous area, and block division was allowed in all frames of the sequence. Our best result was achieved using this method variation, producing a recognition rate of 91.4%.

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

The proposed method has some limitations that shall be addressed in the future. In the second variation, the background treatment did not generate the desired results. Different approaches for this kind of treatment should be explored in order to improve the method. Another issue is related to the frame division into blocks. When we divide a frame, if the frame's width and height are not divisible by the block size, the remaining area on the right and bottom of the first frame of a trajectory are not considered. We believe some important movement may be present in this area. So, we could also adapt our descriptor to take this information into account.

REFERENCES

AMEL, A. M.; ABDESSALEM, B. A.; ABDELLATIF, M. Video shot boundary detection using motion activity descriptor. **Journal of Telecommunications**, v. 2, n. 1, p. 54–59, 2010.

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. In: **Computer vision–ECCV 2006**, 2006. p. 404–417.

CAETANO, F. A. **A Video Descriptor using Orientation Tensors and Shape-based Trajectory Clustering**. Dissertação (Mestrado) — UNIVERSIDADE FEDERAL DE JUIZ DE FORA, 2014.

CHAMBERS, G. S.; VENKATESH, S.; WEST, G. A.; BUI, H. H. Hierarchical recognition of intentional human gestures for sports video annotation. In: IEEE. **Pattern Recognition, 2002. Proceedings. 16th International Conference on**, 2002. v. 2, p. 1082–1085.

CHUNG, J.; KIM, J. R.; SHIM, K. Vision based motion tracking system for interactive entertainment applications. In: IEEE. **TENCON 2005 2005 IEEE Region 10**, 2005. p. 1–6.

FIGUEIREDO, A. M.; MAIA, H. A.; OLIVEIRA, F. L.; MOTA, V. F.; VIEIRA, M. B. A video tensor self-descriptor based on block matching. In: **Computational Science and Its Applications–ICCSA 2014**, 2014. p. 401–414.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. **Communications of the ACM**, ACM, v. 24, n. 6, p. 381–395, 1981.

HAFIANE, A.; PALANIAPPAN, K.; SEETHARAMAN, G. Uav-video registration using block-based features. In: **IEEE International Geoscience and Remote Sensing Symposium (IGARSS)**, 2008. v. 2, p. 1104–1107.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: CITESEER. **Alvey vision conference**, 1988. v. 15, p. 50.

HOAI, M.; ZISSERMAN, A. Action recognition from weak alignment of body parts. **IEEE PAMI**, v. 32, n. 9, p. 1627–1645, 2010.

JAIN, J. R.; JAIN, A. K. Displacement measurement and its application in interframe image coding. **Communications, IEEE Transactions on**, IEEE, v. 29, n. 12, p. 1799–1808, 1981.

JAIN, M.; JÉGOU, H.; BOUTHEMY, P. Better exploiting motion for better action recognition. In: IEEE. **Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on**, 2013. p. 2555–2562.

JI, Y.; SHIMADA, A.; TANIGUCHI, R.-i. A compact 3d descriptor in roi for human action recognition. In: **IEEE TENCON**, 2010. p. 454–459.

KANADE, B. D.; LUCAS, T. et al. An iterative image registration technique with an application to stereo vision. In: **IJCAI**, 1981. v. 81, p. 674–679.

KLÄSER, A.; MARSZAŁEK, M.; SCHMID, C. A spatio-temporal descriptor based on 3d-gradients. In: **British Machine Vision Conference (BMVC)**, 2008. p. 995–1004.

KOBAYASHI, T.; OTSU, N. Motion recognition using local auto-correlation of space–time gradients. **Pattern Recognition Letters**, Elsevier, v. 33, n. 9, p. 1188–1195, 2012.

LALLÉE, S.; LEMAIGNAN, S.; LENZ, A.; MELHUISH, C.; NATALE, L.; SKACHEK, S.; ZANT, T. van D.; WARNEKEN, F.; DOMINEY, P. F. Towards a platform-independent cooperative human-robot interaction system: I. perception. In: IEEE. **Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on**, 2010. p. 4444–4451.

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**, v. 4, n. 4, p. 438–442, 1994.

LIN, Y.-C.; HU, M.-C.; CHENG, W.-H.; HSIEH, Y.-H.; CHEN, H.-M. Human action recognition and retrieval using sole depth information. In: ACM. **Proceedings of the 20th ACM international conference on Multimedia**, 2012. p. 1053–1056.

LIU, J.; LUO, J.; SHAH, M. Recognizing realistic actions from videos in the wild. In: IEEE. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2009. p. 1996–2003.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**, 1967. v. 1, n. 14, p. 281–297.

MARSZALEK, M.; LAPTEV, I.; SCHMID, C. Actions in context. In: IEEE. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2009. p. 2929–2936.

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**, v. 31, p. 85–91, 2013.

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: IEEE. **Conference on Graphics, Patterns and Images (SIBGRAPI)**, 2012. p. 298–301.

MOTA, V. F.; SOUZA, J. I.; ARAÚJO, A. d. A.; VIEIRA, M. B. Combining orientation tensors for human action recognition. In: IEEE. **Conference on Graphics, Patterns and Images (SIBGRAPI)**, 2013. p. 328–333.

MURTHY, O.; GOECKE, R. Ordered trajectories for large scale human action recognition. In: IEEE. **Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on**, 2013. p. 412–419.

NETRAVALI, A. N.; ROBBINS, J. D. Motion-compensated television coding: Part i. **Bell System Technical Journal**, Wiley Online Library, v. 58, n. 3, p. 631–670, 1979.

NIE, Y.; MA, K.-K. Adaptive rood pattern search for fast block-matching motion estimation. **IEEE Transactions on Image Processing**, v. 11, n. 12, p. 1442–1449, 2002.

ONOFRI, L.; SODA, P. Combining video subsequences for human action recognition. In: IEEE. **Pattern Recognition (ICPR), 2012 21st International Conference on**, 2012. p. 597–600.

PEREZ, E. d. A.; MOTA, V. F.; MACIEL, L. M.; SAD, D.; VIEIRA, M. B. Combining gradient histograms using orientation tensors for human action recognition. In: IEEE. **21st International Conference on Pattern Recognition (ICPR)**, 2012. p. 3460–3463.

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**, v. 6, n. 3, p. 313–317, 1996.

RAUTARAY, S. S.; AGRAWAL, A. Design of gesture recognition system for dynamic user interface. In: IEEE. **Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on**, 2012. p. 1–6.

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: IEEE. **Conference on Graphics, Patterns and Images (SIBGRAPI)**, 2013. p. 70–74.

SCHULDT, C.; LAPTEV, I.; CAPUTO, B. Recognizing human actions: a local svm approach. In: IEEE. **Proceedings of the 17th International Conference on Pattern Recognition (ICPR)**, 2004. v. 3, p. 32–36.

SUN, J.; WU, X.; YAN, S.; CHEONG, L.-F.; CHUA, T.-S.; LI, J. Hierarchical spatio-temporal context modeling for action recognition. In: IEEE. **Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on**, 2009. p. 2004–2011.

SUN, X.; DIVAKARAN, A.; MANJUNATH, B. A motion activity descriptor and its extraction in compressed domain. In: **Proceedings of the 2nd IEEE Pacific Rim Conference on Multimedia**, 2001. p. 450–457.

TABIA, H.; GOUIFFES, M.; LACASSAGNE, L. Motion histogram quantification for human action recognition. In: IEEE. **Pattern Recognition (ICPR), 2012 21st International Conference on**, 2012. p. 2404–2407.

VRIGKAS, M.; KARAVASILIS, V.; NIKOU, C.; KAKADIARIS, I. A. Matching mixtures of curves for human action recognition. **Computer Vision and Image Understanding**, Elsevier, v. 119, p. 27–40, 2014.

WANG, C.; WANG, Y.; YUILLE, A. L. An approach to pose-based action recognition. In: IEEE. **Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on**, 2013. p. 915–922.

WANG, H.; KLÄSER, A.; SCHMID, C.; LIU, C.-L. Dense trajectories and motion boundary descriptors for action recognition. **International Journal of Computer Vision**, v. 103, n. 1, p. 60–79, 2013.

WANG, H.; SCHMID, C. et al. Action recognition with improved trajectories. In: **International Conference on Computer Vision**, 2013.

YI, Y.; LIN, Y. Human action recognition with salient trajectories. **Signal processing**, Elsevier, v. 93, n. 11, p. 2932–2941, 2013.

YU, X.; YANG, S. X. A study of motion recognition from video sequences. **Computing and Visualization in Science**, Springer, v. 8, n. 1, p. 19–25, 2005.

YUAN, F.; XIA, G.-S.; SAHBI, H.; PRINET, V. Mid-level features and spatio-temporal context for activity recognition. **Pattern Recognition**, Elsevier, v. 45, n. 12, p. 4182–4191, 2012.

ZHU, S.; MA, K.-K. A new diamond search algorithm for fast block-matching motion estimation. **IEEE Transactions on Image Processing**, v. 9, n. 2, p. 287–290, 2000.