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

Felipe Andrade Caetano

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

Juiz de Fora

2014

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

Felipe Andrade Caetano

A Video Descriptor using Orientation Tensors and Shape-based 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: Rodrigo Luis de Souza da Silva
Coorientador: Marcelo Bernardes Vieira

Juiz de Fora
2014

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

Caetano, Felipe Andrade.
A Video Descriptor using Orientation Tensors and Shape-based Trajectory Clustering / Felipe Andrade Caetano. -- 2014.
68 p. : il.

Orientador: Rodrigo Luis de Souza da Silva
Coorientador: Marcelo Bernardes Vieira
Dissertação (mestrado acadêmico) - Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas. Programa de Pós-Graduação em Ciência da Computação, 2014.

1. Dense trajectories. 2. Human action recognition in videos. 3. Self-descriptor. 4. Orientation Tensor. 5. Clustering. I. Silva, Rodrigo Luis de Souza da, orient. II. Vieira, Marcelo Bernardes, coorient. III. Título.

Felipe Andrade Caetano

A Video Descriptor using Orientation Tensors and Shape-based 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 29 de Agosto de 2014.

Ao meu tio José Americo.

RESUMO

Trajetórias densas têm se mostrado um método extremamente promissor na área de reconhecimento de ações humanas. Baseado nisso, propomos um novo tipo de descritor de vídeos, calculado a partir da relação do fluxo ótico que compõe a trajetória com o gradiente de sua vizinhança e sua localidade espaço-temporal. Tensores de orientação são usados para acumular informação relevante ao longo do vídeo, representando tendências de direção do descritor para aquele tipo de movimento. Além disso, um método para aglomerar trajetórias usando o seu formato como métrica é proposto. Isso permite acumular características de movimentos distintos em tensores separados e diferenciar com maior facilidade trajetórias que são criadas por movimentos reais das que são geradas a partir do movimento de câmera. O método proposto foi capaz de atingir os melhores níveis de reconhecimento conhecidos para métodos com a restrição de métodos autodescritores em bases populares — Hollywood2 (Acima de 46%) e KTH (Acima de 94%).

Palavras-chave: Trajetórias Densas. Reconhecimento de ações humanas em vídeos. Autodescritor. Tensor de orientação.

ABSTRACT

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

Keywords: Dense trajectories. Human action recognition in videos. Self-descriptor. Orientation tensor. Clustering.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\vec{v}$	Vector
a	Scalar
S	Set
$\mathbf{p}$	Point
$\boldsymbol{f}$	Function
$\boldsymbol{M}$	Matrix
$\mathbf{T}$	Tensor

LIST OF ACRONYMS

BoF	Bag-of-Features
BoW	Bag-of-Words
HOG	Histogram of Oriented Gradients
HOF	Histogram of Optical Flow
MBH	Motion Boundaries Histogram
RANSAC	Random Sample Consensus
SSD	Sum of Square Differences
SURF	Speed Up Robust Features
SVM	Support Vector Machine
VLAD	Vector of Locally Aggregated Descriptors
VLAT	Vector of Locally Aggregated Tensors

CONTENTS

1 INTRODUCTION

The human action recognition is a challenging problem in the computer vision that aims to automatically label which action is being performed by a human in a given video. Due to the wideness of its potential applications, this task has been drawing lots of attention and is interest of study in the last two decades.

Most of the main methods to address this problem is focused in extracting lots of small "visual words" (also called features) from the video and merging all the information together using a machine learning step. This approach is inherited from the image classification problem and even some of the most common features extractor methods can be seen as an extension of the image case, like Scovanner et al. (2007), Klaser et al. (2008), Willems et al. (2008) and Laptev (2005). This kind of descriptor generalizes the problem of finding interesting features in the space to the space-time coordinates.

Among other space-time strategies, dense trajectories (WANG et al., 2011) and its related works (JIANG et al., 2012) (JAIN et al., 2013) (WANG; SCHMID, 2013) has shown the greatest recognition rates in the area, therefore being one of the most promising methods to address this problem. The dense trajectories method works by extracting stable points of each frame and tracking them in the subsequent frames, using an optical flow algorithm. Each one of these displacement vectors provided by the optical flow becomes a trajectory when concatenated. Many features like the histogram of oriented gradients (HOG)(DALAL; TRIGGS, 2005), histogram of optical flow (HOF)(LAPTEV et al., 2008) and motion boundaries histogram (MBH)(DALAL et al., 2006) are extracted along the trajectory.

While the process of gathering lots of small visual words together using a machine learning step seems to achieve a very good recognition rate in most works, we rather not use it. The computed final descriptor for each video is based on information that is assembled using all features from all videos in the training database. We believe that is possible to achieve good recognition rates using only the information provided by the video itself — namely self-descriptor.

There are already good methods to achieve a good recognition rate for some simple databases like *KTH* (SCHULDT et al., 2004) and *Weizmann* (BLANK et al., 2005), for example. But most of its videos are made in a synthetic scenario with controlled environments. When taking into account more complex databases like *Hollywood2* (MARSZALEK et al., 2009) and *HMDB51* (KUEHNE et al., 2011) which are based, respectively, in scenes extracted from Hollywood movies and personal public video recordings, we have a different scenario, because even modern methods have some difficulties dealing with intrinsic computer vision problems like bright variation, camera movement and other problems discussed in the Section 1.4 and presented in both databases.

1.1 MOTIVATION

Human action recognition in videos has countless potential applications. There could be mentioned, among others:

- Automatic Surveillance Systems: nowadays, most of the surveillance systems still lie on the human capability to look for a set of cameras and find suspect activities. Such systems, not only are too expensive (as you need to pay a dedicated employee just to fulfill that task) but, are also a point of failure because it is based in the human visual perception. Humans can perform a limited number of tasks and pay attention to a limited number of things at the same time, whereas a computer with enough processing capabilities can process all the data uninterruptedly;

- Automatic content-based video retrieval: the recent growth of the internet along with cheaper digital portable cameras and social media popularity, are leading to a massive increase in the number of videos published by the users on the web. Along with that, comes the problem of indexing and retrieving such content. Nowadays, most of the videos are retrieved using annotation labels, or keywords, usually made by the video owner itself. However, a better approach to address such problem could by performed by a system that automatically find out actions performed in the video, without any user interference. Thus, solving problems like missing keywords and different keywords to describe the same action;

- Human-computer interface: over the years and with the technology progress, the human-computer interaction systems have been improved as well. Nowadays, there are systems capable of recognizing gestures and facial expression satisfactorily. However, this is still an area under development and modern researches in the human action recognition area might be used as a base to improve those interaction systems, and make them faster and easier for humans;

- Children and elderly monitoring: a system able to recognize human actions could be used to support and monitoring children or elderly persons in a smart house. Such system could be a helping hand to avoid domestic accidents and simplify daily tasks.

1.2 PROBLEM DEFINITION

Given a set of videos from a database and a set of human actions, the problem is to find a vector of m coefficients (or descriptor) $\vec{v} \in V \subset \mathbb{R}^m$. The vector $\vec{v}$ must summarize relevant information and best describe the action performed for each video.

It is desirable to have the smallest value for m that is still able to carry information about the action performed in the video. The smallest the dimension of the descriptor is, the more is its power to sum up information. Ideally, if two different videos represent the same action, then the distance between each other in the space V should be small and if two videos represent different actions, their distance should be proportional to the actions similarity. In real scenarios this may not always happen as descriptors from different actions are not always linearly separable.

One of the restrictions of this work is that the method must be capable of creating a descriptor for the video using only the information given by the video itself. In other words, the method cannot look up for any other video information in the same database, as it is done in the bag-of-features (BoF) (SIVIC; ZISSERMAN, 2003) and similar approaches. Methods subjected to this restriction are classified as self-descriptor.

1.3 OBJECTIVES

The main objective of this work is to create a method to approach the human action recognition problem, generating a new descriptor combining dense trajectories proposed by Wang et al. (2011) with the tensor-based approach used by Perez et al. (2012). Our hypothesis is that this new descriptor is capable of merging both movement information, given by the dense trajectories Wang et al. (2011) and shape information from the HOG (DALAL; TRIGGS, 2005) in a very simple but effective way.

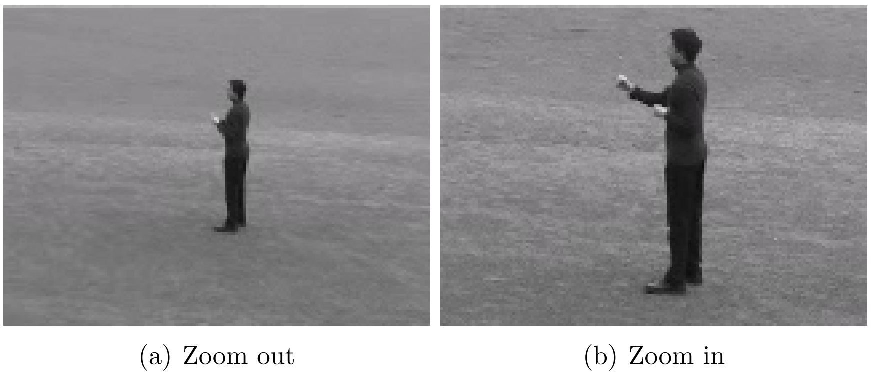
The secondary objective is to propose a method to cluster trajectories based on their shape, similar to what is proposed by Jiang et al. (2012). This method is, additionally, capable of clustering together trajectories that have the same shape, but different directions. Doing so, we can estimate which trajectories are generate by camera movement and separate them in a different cluster.

1.4 CHALLENGES

Just like any other field in the computer vision research area, the human action recognition has some intrinsic challenges that should be addressed in order to maximize the useful information extracted. For example:

- Multiple actions in one single video: one video may have more than two actions performed in the same take. This make the task to recognizing actions even more challenging. The descriptor for that video must summarize the information of both actions simultaneously. An example is shown in the Figure 1.1, where a person is driving a car and answering the phone at the same time;
- Point of view variation: the method should be able to recognize an action in despite of variation in the camera's point of view due to movement or zoom. For example, Figure 1.2 shows the impact of changing the camera's zoom between two near frames of the same video;
- Ambiguous actions: some actions might seem very similar but, they are different. For instance, in the *KTH* Database, both *running* and *jogging* videos are very similar and sometimes, even a human could not be able to distinguish them, as shown in the Figure 1.3. This is also called as inter-class similarity;
- Different representations of the same action: one action may be executed in lots of different ways. For example, the "eat" action in the Hollywood 2 can be performed in many different situations, with different features. This is also called as intra-class variation. Another example is given by the Figure 1.4, where the same action is performed in two different conditions, thus, is harder to match features between those videos;

Figure 1.1: Scene in the Hollywood 2 database where a man drives a car while answering the phone.



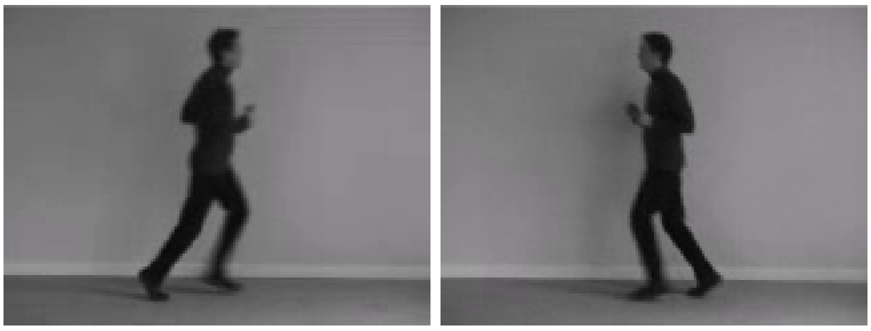
(a) Zoom out (b) Zoom in

Figure 1.2: In (a) and (b) the same action performed with zoom variation, leading to a change in the scale on both frames.

- Illumination: the variation in the illumination of the scene causes a high change in the pixel values. For example, in the Figure 1.5, the values of intensity in most of the pixels are very different because of the scene's luminosity. A good method should be able to handle those conditions;
- Partial occlusion: sometimes, an action may happen when it is not fully on the camera's frame or may be partially occluded by some object. For example, if one wants to detect the action of sitting up, it is possible that the movement is not fully captured by the camera or there's a table in the way. But yet, it is still notable and should be detected. Another example is shown in the Figure 1.6.

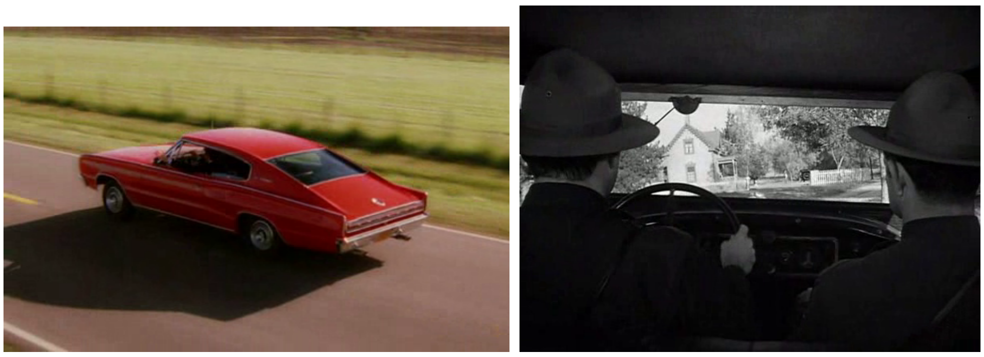
1.5 OUTLINE

The remainder of this work is organized as follows: the Chapter 2 presents works that are related to our proposed method and classical approaches to address the same problem.



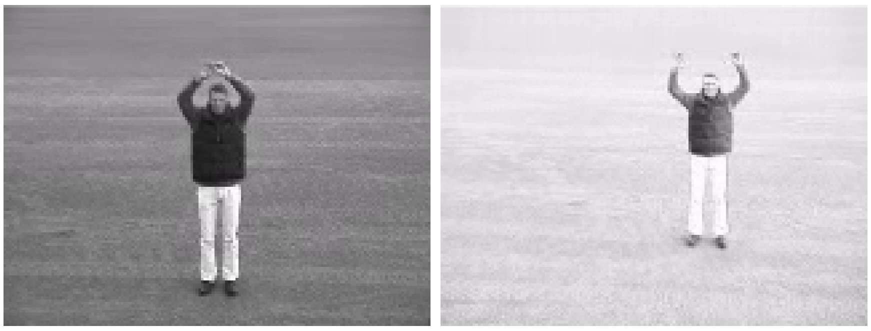
(a) Running (b) Jogging

Figure 1.3: Different actions that look alike in the KTH database.



(a) Driving action from outside the car's perspective (b) Driving action from inside the car's perspective

Figure 1.4: In (a) and (b), the same action is represented in very distinct ways.



(a) Darker scene (b) Lighter scene

Figure 1.5: Same action performed by the same person but with different lighting conditions in the KTH database.

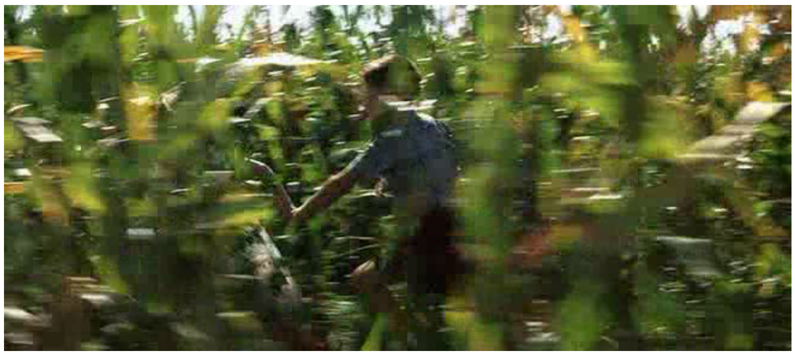


Figure 1.6: Scene in the Hollywood 2 database where a kid is running in a field. The plants are partially hiding the action.

In the Chapter 3 we present the fundamentals that contribute to the foundations of this work. In the Chapter 4 we present a detailed description of our contributions and how they are merged together in our method. In the Chapter 5 we present our experimental results and the comparison with known methods in the area. In the Chapter 6 we present our conclusion remarks and discuss the possible future works.

2 RELATED WORKS

In this chapter, we present similar and relevant works in the area of human action recognition that contributed in our work.

2.1 TENSOR BASED APPROACHES

Tensors are a powerful and robust tool that has been used in the last decade in many different works in the pattern recognition area. In computer vision area, tensors are largely used to compute optical flow. Recent works were proposed to use tensors as a way to gather large amounts of data, keeping most of the relevant information and, mainly, the relationship between them.

Yuan et al. (2010) created a new descriptor based on the covariance matrix of three concatenated low-level features: the position of the pixel in space and time, the gradient vector and the optical flow vector. These features are extracted on a spatio-temporal cuboid selected using the method proposed by Dollar et al. (2005). The authors argue that covariance matrices do not lie on Euclidean space, thus they apply a Log-Euclidean Riemann metric to measure the distance between those visual words and later they are quantized to form an appearance dictionary, like it is done in the BoF approach. A similarity between this work and ours is that we also use a covariance matrix to encode our descriptor. However, we not only use it just to represent the correlation between the coefficients of the original vector, but as a tensor that accumulates information along the whole video. Another similarity is that the covariance matrix in their proposed method is generated from a concatenation between vectors that holds different source of information. We also use this idea as we concatenate a histogram with a position point (x, y, t) before generating the tensor.

Perez et al. (2012) proposed a method to address the human action recognition problem combining HOG3D (KLASER et al., 2008) and orientation tensors. Each frame of the video is divided into equally sized windows and a tensor is computed for each one of the windows. After that, the tensor of the frame is computed by the sum of the normalized tensors of each window. The final descriptor is the sum of all frame's tensor. The method shows a fairly competitive recognition rate, in despite of using a global approach (no

Interest Points involved — all pixels are used to generate the final descriptor). Just like our method, the final descriptor in his approach is also calculated using only data provided by the video itself. Another similarity between his work and ours is that we both use orientation tensors to encode the final descriptor.

Mota et al. (2012) used the projection of the optical flow on Legendre polynomials to find its coefficients. An orientation tensor based on the computed coefficients is calculated for each frame and accumulated through the entire video. This method uses a global approach and enhances the power to combine information of the orientation tensor. The authors extols the fact that the use of tensors lead to an descriptor that is independent of the frame size and the length of the video, which is good for databases that include more realistic videos.

Mota et al. (2013) also used orientation tensors to encode a descriptor based on HOG3D. But, the author argues that aggregating several tensors over the time, as done in (PEREZ et al., 2012), could lead to an isotropic tensor, which is a tensor that does not represent a direction of the information — or all the directions equally. Thus, they propose a method to improve the descriptor by changing the way the aggregation of the tensors is done. In their approach, the video is split in equally sized blocks and the orientation tensor computed over the HOG3D is calculated for each block. The final descriptor is the result of the concatenation of those blocks.

Sad et al. (2013) extended the work proposed by Perez et al. (2012) by combining multiple first order derivative estimators of the gradient in a single descriptor. The author argue that by using multiple filters it is possible to capture different motion nuances that characterize the movement and thus a better representation of the action performed. The final descriptor is calculated just as it is in (PEREZ et al., 2012), but it is done multiple times, using each derivative estimator at a time and concatenating all tensors in the end.

Mota et al. (2014) proposed a method to address the human recognition problem combining what is proposed by Mota et al. (2012) and Perez et al. (2012). The final descriptor is the concatenation of the tensors generated by both methods. The author shows that aggregating different source of information provided by the HOG and the optical flow descriptors may enhance the recognition rates in most of popular databases.

Picard and Gosselin (2011) improved the approach presented in (JEGOU et al., 2010) by representing the descriptors as orientation tensors and aggregating them together around the same center. Although their work is focused in the content based image retrieval problem, this is still a good example of how using tensors to accumulate information can outperform other approaches based on vectors.

2.2 TRAJECTORY BASED APPROACHES

The dense trajectories proposed by (WANG et al., 2011) showed to be a good method to discriminate local motion information in videos and led to a wide collection of inspired works.

Jiang et al. (2012) used k-means to cluster trajectories using the displacement between the first and last point of the trajectory. This is performed at each five frames and the trajectories are grouped into five clusters. The author assumed that the top three largest clusters are candidates to the dominant motion and use the mean motion of these to compensate other trajectories. After that, the author uses a pairwise motion representation to generate a visual codebook to the standard BoF approach. The similarity between this approach and ours is that we also try to cluster trajectories to find the dominant motion. However, we do not use the displacement of the trajectory and neither the classical trajectory shape definition to cluster trajectories, but a vector with the angle between each two subsequent displacement vectors. We believe that we can group more similar trajectories by doing so and our approach can keep invariability to both scale, translation and rotation changes at the same time. More details will be given in the Chapter 4

Jain et al. (2013) improved the dense trajectory approach detailed in Section 3.7 by decomposing visual motion into the residual and dominant motion using an affine model. The dominant motion is assumed to be due to the camera's motion, while the residual motion is assumed to be due to the independent scene motions. By that, the author tried to compensate the camera movement in the calculation of the optical flow and reinforce the focus in the action performed. However, the author also showed that the camera motion could not be thrown away, as it is useful as complementary information to recognize some action categories. We use a similar idea in our approach, as we also try to separate the trajectories made by the camera movement and also keep it as additional information. In his work he proposed a new descriptor called Divergence-Curl-Shear that computes local kinematic features of the optical flow along the trajectory. Unlike our approach, the final descriptor is computed using VLAD, an extension of the BoF.

Wang and Schmid (2013) is the current state-of-art in the field. The authors improved the dense trajectories by estimating the camera's motion and correcting it. They used SURF (BAY et al., 2008) to find features points and match them in two consecutive frames. With that information, along the dense trajectories itself, they are capable of using RANSAC (FISCHLER; BOLLES, 1981) to estimate the homography between those frames and assume that the transformation in made by the camera motion. To ensure that the movement done by the human(s) in the action will not interfere in the above steps, they use a human detector (PREST et al., 2012) and ignored the information extracted around those areas. After calculating the camera movement, the optical flow and, consequently, the trajectories are compensated to minimize the camera's movement influence in the extracted features. To extract the features, those areas around humans previously ignored are now taking into account and the same "visual words" extracted in (WANG et al., 2011) are now extracted around those trajectories. The final descriptor is computed using Fish Vectors (PERRONNIN et al., 2010) and the standard BoF approach.

As we can see, most of the works are focused in eliminate or compensate the camera's motion. We apply the same idea here, but in a very simpler approach; no complex methods to detect the camera's motion is applied. Most of these recent works are also interested in using a BoF approach to form the video's descriptor. We are not interested in applying it here; since we want the descriptor to be computed using only the information provided by that video.

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

3 FUNDAMENTALS

In this chapter we present the main fundamentals of this work.

3.1 HISTOGRAM OF ORIENTED GRADIENTS

HOG is a method proposed by Dalal and Triggs (2005) originally designed to address the human recognition problem. The HOG has the power to summarize information from a region, keeping information about its shape and appearance while achieving some illumination invariance. HOG is basically a description about the distribution of the edge directions in an image or region of an image.

The gradient is a vector that points to direction of maximum change in the intensity of bright of a pixel $\mathbf{p}$ and is denoted by

$$\nabla \vec{f}(\mathbf{p}) = \left(\frac{\partial \boldsymbol{f}(\mathbf{p})}{\partial x}, \frac{\partial \boldsymbol{f}(\mathbf{p})}{\partial y} \right) = (I_x, I_y) \, .$$

As we are dealing with a discrete space, the partial derivatives are approximated using a central derivative filter. In order to extract the histogram of the gradients, they are transformed to the polar coordinates system using

$$r = \sqrt{(I_x)^2 + (I_y)^2}$$

and

$$\Theta = \arcsin\left(\frac{I_y}{I_x}\right) ,$$

where Θ determine the orientation of the gradient, r its magnitude and I_a is the partial derivative in respect to a.

After all gradients of the region are computed, they are quantized in a histogram. The vote of the gradient to a bin is its magnitude. Gradients close to the edges of the block are downweighted with a Gaussian function. The above steps are summarized in the Figure 3.1. We can clearly see that the final histogram does not keep all the information about the gradients but, it has the power to put together information about all gradients in a single and smaller representation.

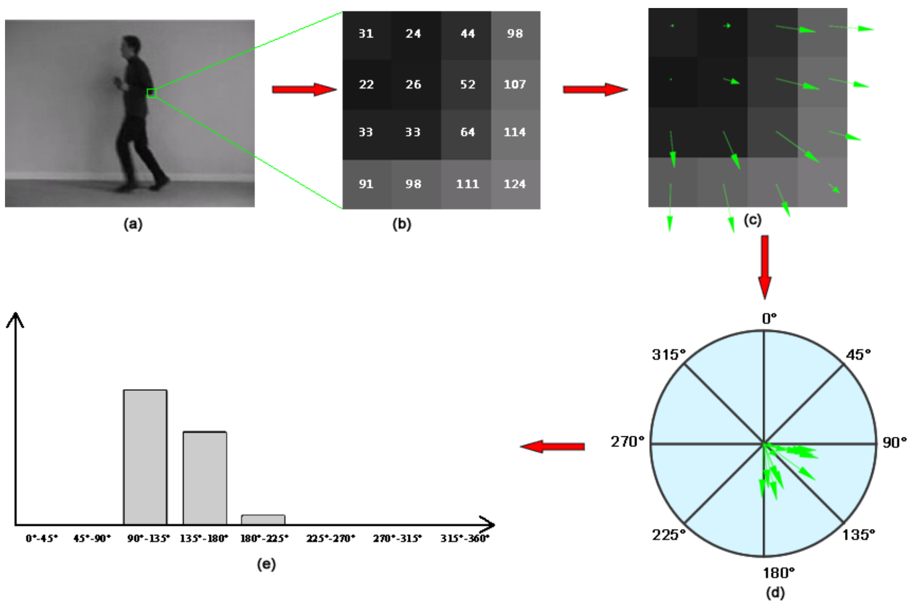


Figure 3.1: Steps to compute the HOG. (a) The original image is sampled. (b) The brightness of each pixel in the window. (c) The gradient of each pixel, pointing to the direction of greatest change of brightness. (d) Each gradient of the window quantized in a histogram using the polar coordinate system. (e) The final histogram.

Klaser et al. (2008) proposed an extension to the classical HOG method, the HOG3D. This new method also takes into account the derivatives in the time dimension. That said, the gradient now is defined as

$$\nabla \boldsymbol{f}(\mathbf{p}) = \left(\frac{\partial \boldsymbol{f}(\mathbf{p})}{\partial x}, \frac{\partial \boldsymbol{f}(\mathbf{p})}{\partial y}, \frac{\partial \boldsymbol{f}(\mathbf{p})}{\partial t} \right) = (I_x, I_y, I_t),$$

or transforming to spherical coordinates:

$$r = \sqrt{(I_x)^2 + (I_y)^2 + (I_t)^2},$$

with

$$\theta = \arccos\left(\frac{I_t}{r}\right)$$

and

$$\phi = \arctan\left(\frac{I_y}{I_x}\right),$$

where $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$. To quantize the final histogram, Klaser et al. (2008) proposed to approximate the bins of the histogram to a polyhedron with congruent faces, or platonic solid as described in the Figure 3.3. A straighter way to quantize the gradients is to divide θ and ϕ into equally spaced bins. This approach leads us to a mapping similar

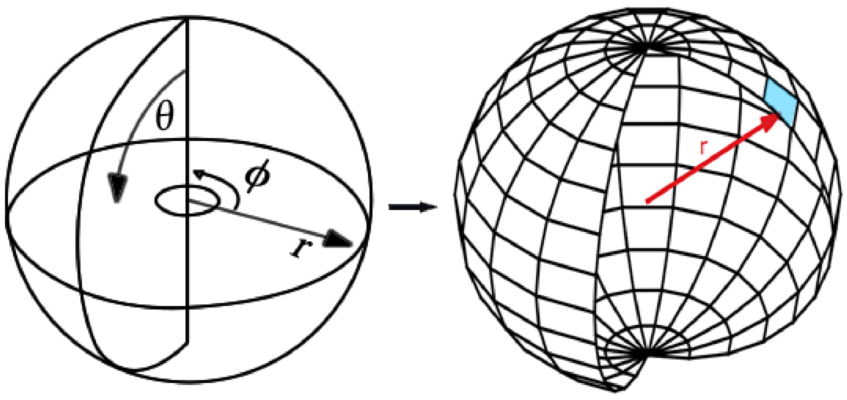
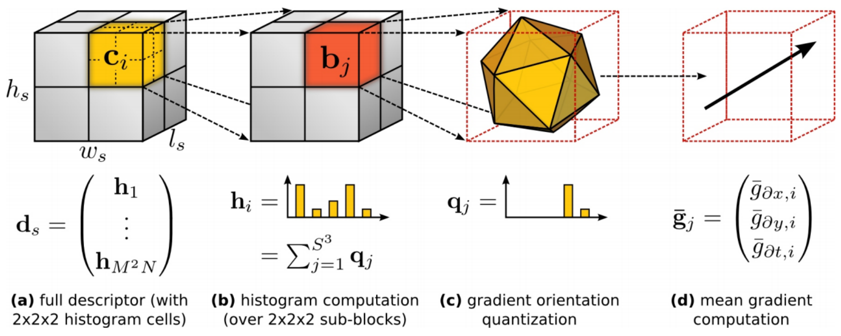


Figure 3.2: An illustration of how bins are quantized using spherical coordinates. In the left, a representation of how a vector can be positioned. In the right a representation of how equally spaced bins looks.



h_s
w_s
l_s
$\mathbf{c}_i$
$\mathbf{b}_j$
$\mathbf{d}_s = \begin{pmatrix}\mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{M^2N}\end{pmatrix}$
$\mathbf{h}_i =$
$= \sum_{j=1}^{S^3} \mathbf{q}_j$
$\mathbf{q}_j =$
$\bar{\mathbf{g}}_j = \begin{pmatrix}\bar{g}_{\partial x,i} \\ \bar{g}_{\partial y,i} \\ \bar{g}_{\partial t,i}\end{pmatrix}$
(a) full descriptor (with 2x2x2 histogram cells)
(b) histogram computation (over 2x2x2 sub-blocks)
(c) gradient orientation quantization
(d) mean gradient computation

Figure 3.3: Steps to compute the HO3D (KLASER et al., 2008).

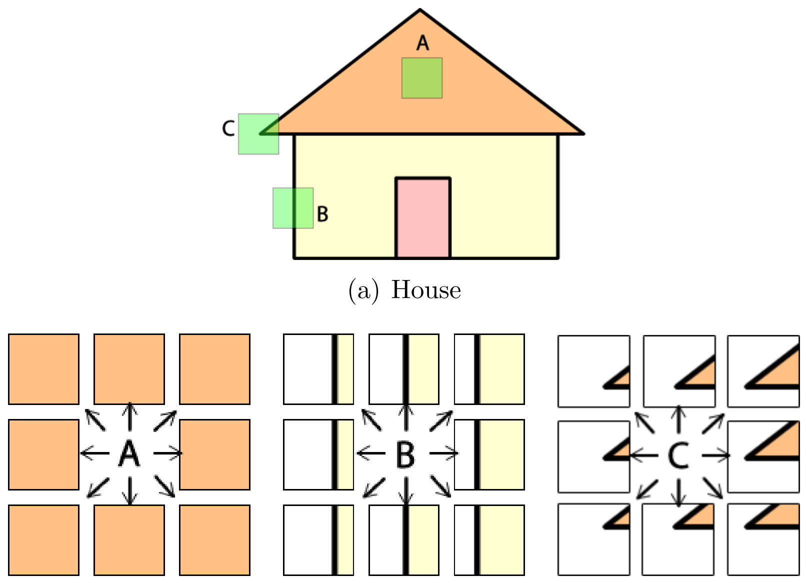
to longitudes and latitudes on a globe. This is a much simpler approach but, there is a problem related with the singularity on the poles of the globe. Using this approach, the resulting histogram size is

$$h_{size} = bins.\left(\frac{bins}{2}\right),$$

where *bins* represents the number of subdivisions in the histogram. This happens because in the spherical coordinate system, the variables to encode the position of a bin θ and ϕ are in $[0, \pi]$ and $[0, 2\pi]$ intervals, respectively. Thus, to keep them equally spaced, θ requires the half of the ϕ subdivisions. The Figure 3.2 shows a representation of the process.

3.2 HARRIS DETECTOR

The Harris detector is a method to extract interest points proposed by Harris and Stephens (1988) based on the principle that corners are good interest points. This is reliable due to



(a) House

(b) Flat region (c) Edge region (d) Corner region

Figure 3.4: An illustration of different region types in an image.

the fact that the corners have a greater resistance to incorrect matching. This property can be illustrated in the Figure 3.4. For both points A (flat region) and B (edge region), there are many other points that would match in the same image, which does not happen with the Region C (corner region).

To visualize this effect one may shift an observation window in every direction around the point and compare the results. In Figure 3.4b, we can see that moving the observation window in any direction in a flat region would lead to an unchanged sampled image. Thus, this is not a stable region to use for matching. In Figure 3.4c, we can see that moving the window in a perpendicular direction of the edge might lead to a change but, when moving along the edge, there is no change in the image. In Figure 3.4d, we can see that any direction taken by the window would still produce changes in the sampled image and that is why corners are commonly chosen as interest points. This property is extremely correlated with the aperture problem addressed in the Section 3.3.

In order to mathematically distinguish corners from other points in images, Harris and Stephens (1988) proposed the use of the auto-correlation, which is in the form:

$$\boldsymbol{f} \otimes \boldsymbol{f} = \sum_{I} \sum_{J} \boldsymbol{f}(x, y) \boldsymbol{f}(x + i, y + j),$$

where $\boldsymbol{f}$ is the function of the image, (x, y) is the index of each pixel in window around the analyzed point and (i, j) is the displacement of the window.

The auto-correlation returns a high value if the displaced image is similar to the original image in the (i, j) direction and a low value otherwise. So, if the point is in a flat region, the auto-correlation is high in any direction (i, j) around the point. If the point is in an edge region, there is just one direction in which the correlation is always high. If the point is in a corner, the auto-correlation should not be high in any direction. Given that and using the fact that minimizing the auto-correlation has the same effect of maximizing the sum of square differences (SSD), Harris and Stephens (1988) defined the function

$$\boldsymbol{e}(u, v) = \sum_{x,y} \boldsymbol{w}(x, y)[I(x + u, y + v) - I(x, y)]^2,$$

where $\boldsymbol{w}$ is a weight function, I is the intensity of the pixel, (x, y) are all possible pixels around the point and (u, v) is the displacement of the window. Applying Taylor series in the $I(x + u, y + v)$ term, we got

$$\boldsymbol{e}(u, v) = \sum_{x,y} \boldsymbol{w}(x, y) \left[uI_x + vI_y\right]^2,$$

where I_x and I_y are derivatives in respect to x and y.

Then, the authors manipulated the equation and got the following result:

$$\boldsymbol{e}(u, v) = \begin{pmatrix} u & v \end{pmatrix} \boldsymbol{M} \begin{pmatrix} u \\ v \end{pmatrix},$$

where

$$\boldsymbol{M} = \sum_{x,y} \boldsymbol{w}(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}.$$

The function $\boldsymbol{e}(u, v)$ can be seen as the equation of an ellipse with covariance matrix $\boldsymbol{M}$. The authors showed that by decomposing the matrix $\boldsymbol{M}$ into its eigenvalues and eigenvectors, one could analyze the direction of maximum change of the function $\boldsymbol{e}$, where each eigenvector gives you the direction of the change and the corresponding eigenvalue, the intensity of change.

Using the property mentioned above, Harris and Stephens (1988) proposed to analyze the eigenvalues of $\boldsymbol{M}$ to define a corner. As depicted in the Figure 3.5, high values to λ_1

and λ_2 indicates a high variation in both directions, and therefore that point is probably in a corner. A high value in just one eigenvalue indicates an edge. And lower values in both λ_1 and λ_2 indicates a flat region.

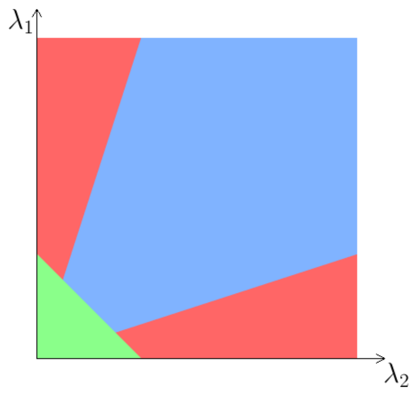


Figure 3.5: Eigenvalues of the matrix $\boldsymbol{M}$. Green means it is flat. Red means it is an edge. And blue means it is a corner.

The authors also proposed the following equation to measure the cornerness using both eigenvalues:

$$r = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2,$$

where k is a constant. If r is a negative value, the region is an edge. If r is positive and small, it is a flat region. If r is high, it is classified as a corner.

3.3 OPTICAL FLOW

The optical flow is an estimation of the bi-dimensional apparent velocities in a sequence of two consecutive frames. In other words, it represents the estimated apparent movement of the scene projected in the camera. Since the basic unit of information is the brightness of each pixel, methods to extract the optical flow have to deal with many problems. For instance, object occlusion, homogeneous regions and large displacements.

There are already lots of proposed methods to compute the optical flow, but this is still an open problem in active research in the computer vision, and no method has yet acquired perfection in the task. Recent algorithms to calculate the optical flow can be found in Baker et al. (2011).

Since one of the basic source of information in a video is the movement, the optical flow is largely used in the human action recognition. One of the most known descriptors, proposed by Laptev et al. (2008) uses the HOF which is computed just like the HOG

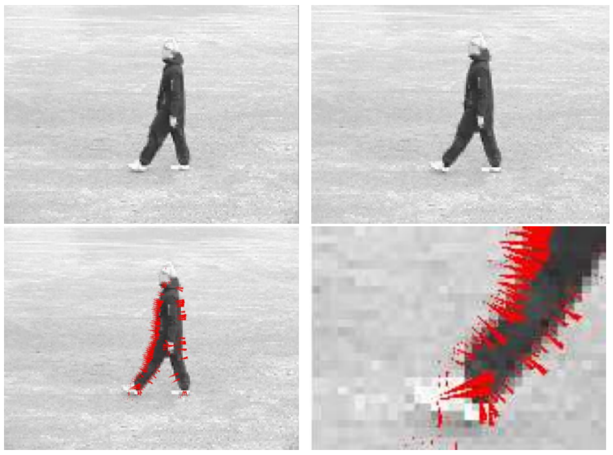


Figure 3.6: Example of the optical flow (MOTA, 2011).

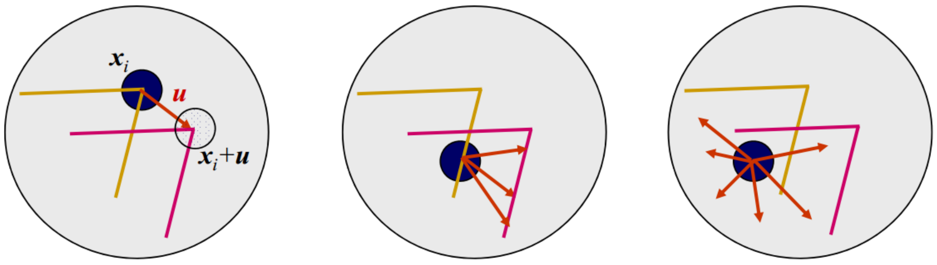


Figure 3.7: An illustration of the aperture problem (SZELISKI, 2010).

presented in the Section 3.1. Another example optical flow used in the human action recognition is the dense trajectory method discussed in the Section 3.7, which uses an optical flow to track points in the video to form trajectories.

One of the greatest challenges in the area is known as the aperture problem, illustrated in the Figure 3.7. For some regions of the image, there is an ambiguity when trying to locally estimate the displacement of a point. We try to overcome this problem using a strategy similar to the Harris detector presented in the Section 3.2.

3.4 K-MEANS

K-means is an unsupervised classification method based on clustering. It works interactively trying to group n samples in k clusters. Initially, each one of the clusters is assigned to a random sample. The other samples are labeled based on the nearest cluster using the Euclidean distance. In every new step, the centroid of the cluster is recalculated and the nearest samples are labeled again, like shown in the Figure 3.8. There are many proposed stopping criteria. The most common is to stop when no sample is labeled with a different

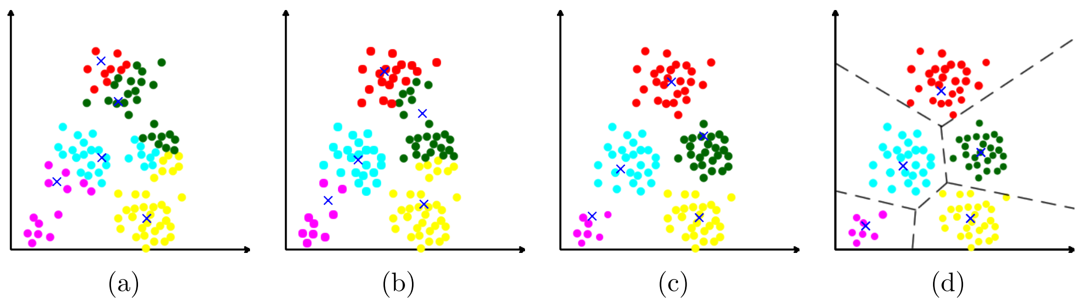


Figure 3.8: Illustration of the K-means algorithm. (a) the centroids are randomly sampled. (b) and (c) the new centroid of each cluster is recomputed and the samples are labeled using the nearest distance.(d) the final result of the clustering and its representation as a Voronoi diagram.

cluster than it had in the previous step.

The k-means method is largely used in computer vision, and it is the base to compute the dictionary of visual words in the BoF approach. The final result of the k-means can be seen as a Voronoi diagram, where any new point is classified according to its nearest centroid.

3.5 BAG-OF-FEATURES

BoF is a strategy inspired in the Bag-of-Words (BoW), initially applied in the content based image retrieval and extended to the human action recognition. The BoW is a method proposed by Baeza-Yates and Ribeiro-Neto (1999) which consists in classifying a document based on a histogram of incidence of its words in a known dictionary, disregarding any semantic or structural characteristic.

A similar idea is applied in the BoF. To compute the dictionary, lots of small "visual words" extracted from the training images or videos are quantized to form a feature space using k-means clustering, a step known as "coding". Each one of the clusters centroids can be seen as a word in the standard BoW method. The final step is the "pooling", where an unknown image or video is represented as its word's frequency. The Figure 3.9 has an illustration of the process. Note that the final descriptor is calculated after lots of information from previous images or videos.

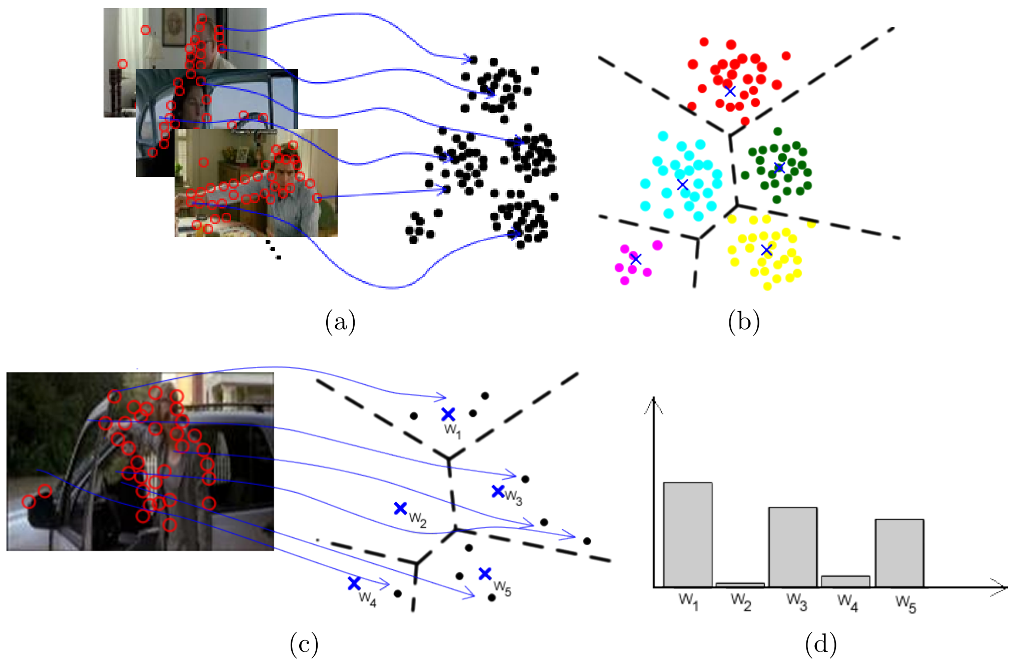


Figure 3.9: Illustration of the BoF steps. (a) all visual words from the training set are gathered together. (b) the visual words are clustered using the k-means algorithm. (c) a new image from the testing set is analyzed and its visual words are matched in the previously generated clusters. (d) the final descriptor is computed based on the frequency of the incidence of each cluster.

3.6 SELF-DESCRIPTOR

Self-descriptor is a constraint about how the final descriptor is generated for the video. As opposed to dictionary-based algorithms for human action recognition, to attend the self-descriptor constraint a method should be able to calculate the final descriptor for the video using only the information provided by the video itself. No outside information can be used in the process of calculating the video's descriptor.

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

Another advantage of using the self-descriptor approach is that it requires much less

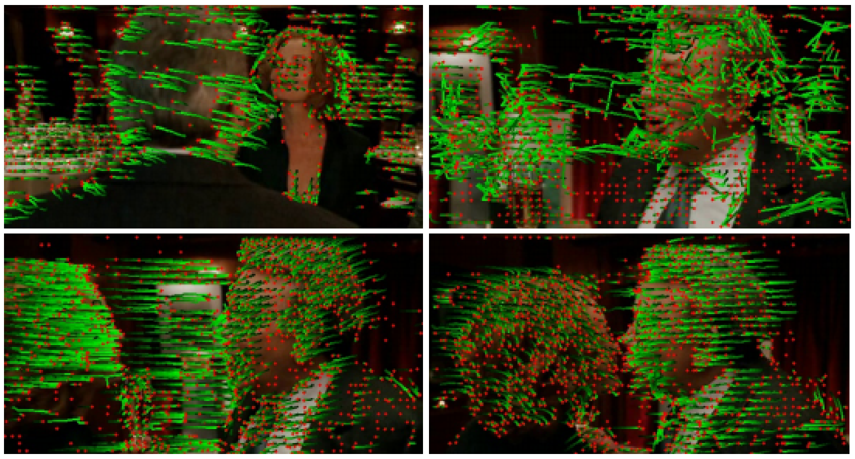


Figure 3.10: Dense trajectories example in a scene from Hollywood2 (WANG et al., 2011).

space in both primary and secondary memory. The secondary memory problem in the BoF often happens because we need all visual words from all the videos to be stored before we can cluster them. A giant amount of space might be necessary. Our experiments in the Hollywood dataset using the default dense trajectories approach required over 700 GB of free space. There is no such problem for the self-descriptor approach as the only space required is the one used by the final descriptor. The primary memory space in non-self-descriptor works becomes a problem when we need to gather all that stored visual words and cluster them together. As we can not just put all of them in the main memory to cluster, we need to use as much space as we can in the main memory and be constantly accessing the secondary memory to use required data.

3.7 DENSE TRAJECTORY

The dense trajectory is a method proposed by Wang et al. (2011) that aims to acquire motion information in videos using displacement vectors provided by the optical flow. The final effect can be seen in the Figure 3.10. Note that there are many other works focused in extracting different trajectories in videos (LUCAS; KANADE, 1981; SUN et al., 2009, 2010) but only the dense trajectory method were detailed in this work, as it is our current choice.

The first step to compute the trajectories is to sample points in the frame using a grid spaced by w pixels. The authors argue that densely sampling points instead of extracting them using any interest point detector results in a better coverage of foreground motion and the surrounding context. The authors also argue that densely sampling shows a better result in presence of fast irregular motions. The sampling is done in different scale spaces. The authors proposed the use of $w = 5$ and a scale factor of $^{1}/_{\sqrt{2}}$, which means that each space scale has the half area of the previous scale.

To avoid points in homogeneous areas and reduce the impact of errors in the optical flow computation, the sampled points are analyzed using a criterion similar to the Harris detector presented in the Section 3.2. The authors proposed a threshold r to eliminate those points, calculated by the formula

$$r = q. \max_{\mathbf{p} \in F} \min(\lambda^1_{\mathbf{p}}, \lambda^2_{\mathbf{p}}),$$

where q is a constant, $\lambda^1_{\mathbf{p}}$ and $\lambda^2_{\mathbf{p}}$ are the eigenvalues of the structure tensor of each pixel $\mathbf{p}$ in the frame F where the points are being sampled. Authors claim that $q = 0.001$ showed the best results in the experiments, successfully removing homogeneous areas without compromising the density of the sampled points.

The main idea in this criterion is to eliminate points which the smallest eigenvalue times $^{1}/_{q}$ is not at least r, where r is computed over the maximum of the minimum eigenvalues. As we are interested in point with both high $\lambda^1_{\mathbf{p}}$ and $\lambda^2_{\mathbf{p}}$, analyzing only the minimum of each for the pixel is enough. Note that although there is a criterion similar to what is proposed on the Harris detector, this one is based on the max eigenvalue of each frame, and not a fixed value. This approach guarantees that at least one point will be sampled and there is flexibility when defining a threshold, which again, reduces the chances of not sampling points densely enough. The Figure 3.11 shows an example of the points sampled and filtered using this criterion.

To characterize the points extracted for the trajectories, we are going to refer $\mathbf{p}^t_{j,i}$ as the j-th point of the i-th trajectory among the trajectories that start at the frame t. Each point has its (x, y) coordinates. We are going to refer P^t_j as the set of all points from trajectories that started at the frame t and are currently at the frame j.

After the set $P^t_0 = \{\mathbf{p}^t_{0,0}, \mathbf{p}^t_{0,1}, \ldots, \mathbf{p}^t_{0,i}, \ldots, \mathbf{p}^t_{0,n-1}\}$ of n initial points of the t-th frame are chosen, the optical flow is extracted using the algorithm proposed by Farnebäck (2003). The optical flow is computed for all pixels in the frame but, its information will only be used to compose the trajectories of the selected points. As the points were filtered using an approach similar to the Harris detector, the selected points have a high probability to be corners and the optical flow on the selected points are less likely to be affected by the

(a) Original frame

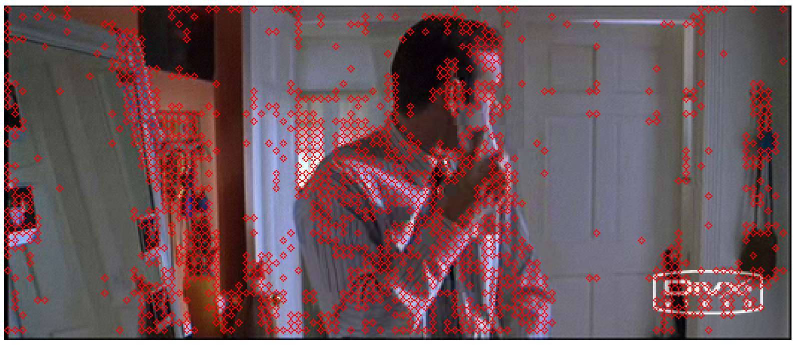

(b) Points densely sampled and filtered

Figure 3.11: An example of the dense trajectories method to extract the initial points.

aperture problem and therefore, they are more reliable.

With the computed optical flow $\Omega_t = (u_t, v_t)$ of the frame t, the next set of points P^t_{j+1} is calculated using P^t_j:

$$P^t_{j+1} = \left\{ \mathbf{p}^t_{j+1,i} \,|\, (x^t_{j+1,i}, y^t_{j+1,i}) = (x^t_{j,i}, y^t_{j,i}) + (\boldsymbol{M} * \Omega_t)|_{x^t_{j,i}, y^t_{j,i}} \right\}, \tag{3.1}$$

where $\boldsymbol{M}$ is a median kernel and i is an index for each initial point. Which means that the next point of the trajectory will be calculated using the displacement of the optical flow in the current point. Filtering the optical flow with a 3×3 median kernel is a good alternative to minimize the impact of any noise. The authors argue this is better than interpolation, because it completely remove noises that otherwise would be just smoothed out.

The next points are calculated using the same Equation 3.1 and after l frames, where l is the size of the trajectory, we have a set of points $\{P^t_0, P^t_1, P^t_2, \ldots, P^t_l\}$ that contains all points from all trajectories that started together at the frame t. The authors proposed a fixed size l to overcome trajectories that drift way from what it was supposed to be. They empirically proposed 15 as a good value to l. With the set of all points, we can define each trajectory as

$$S_i^t = \left\{\mathbf{p}_{0,i}^t, \mathbf{p}_{1,i}^t, \ldots, \mathbf{p}_{l,i}^t\right\},$$

that is the i-th trajectory among the trajectories that started at the frame t. Semantically, the trajectory represents the sequence of the tracked positions of the first initial point.

Trajectories with low displacement are removed in a post-processing stage, because they do not represent any motion information. Similarly, trajectories with sudden high displacement are also removed, because they tend to be erroneous. All these steps to find out trajectories are repeated for each space scale of the video. New points to compose trajectories are extracted at each new frame but, if the point matches with an already existent trajectory, it will not be taken into account to avoid overlapping.

The traditional dense trajectory proposed by Wang et al. (2011) also include a step where visual words are extracted along the trajectories. One of the descriptors is the trajectory itself and is calculated using the set of vectors that connect two points in subsequent frames. This is given by the formula

$$V_t^i = \{\Delta\mathbf{p}_{0,i}^t, \Delta\mathbf{p}_{1,i}^t, \ldots, \Delta\mathbf{p}_{l-1,i}^t\},$$

where $\Delta\mathbf{p}_{j,i}^t = (\mathbf{p}_{j+1,i}^t - \mathbf{p}_{j,i}^t) = (x_{j+1,i}^t - x_{j+1,i}^t, y_{j,i}^t - y_{j+1,i}^t)$. The trajectory descriptor is then normalized with the sum of all vector's length so trajectories representing the same movement in different scales can be matched using

$$\|V_t^i\| = \frac{(\Delta\mathbf{p}_{0,i}^t, \Delta\mathbf{p}_{1,i}^t, \ldots, \Delta\mathbf{p}_{l-1,i}^t)}{\sum_{j=0}^{l-1} \|\Delta\mathbf{p}_{j,i}^t\|}.$$

Also, HOG, HOF and MBH (DALAL et al., 2006) descriptors are extracted along the trajectories. The final visual word is composed by the concatenation of those descriptors for each frame of the trajectory, extracted in a spatio-temporal cuboid. An overview can be seen in Figure 3.12. The final step is to compute a codebook using BoF with those visual words and compute the final descriptor based on the histogram of the visual words. We will not get any deeper in this subject because in this work we only apply the method to extract trajectories and we do not use any of these descriptors, but a new descriptor by our own, proposed in the Section 4.1.

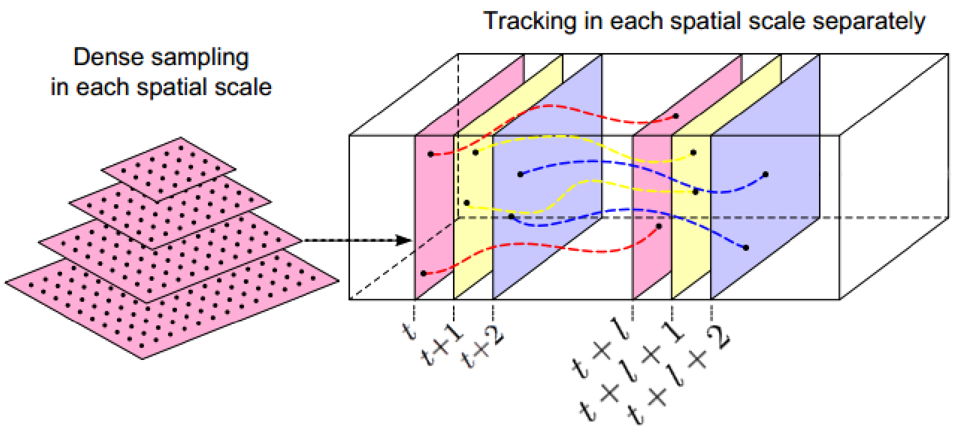


Figure 3.12: Overview of the dense trajectory method. On the left, points are extracted on a grid in each scale. In the right, trajectories are composed by tracking the initial points for l frames using the optical flow (WANG et al., 2011).

3.8 ORIENTATION TENSORS

Tensors are extensions of the vector and matrix concept to higher orders. They have the power to describe relations between vectors, scalars, and other tensors and they are independent of a coordinate system. A special case of tensor is the orientation tensor. It is a square symmetric positive-definite matrix. A vector $\vec{v}$ can be represented as an orientation tensor using $\mathbf{T} = \vec{v}\vec{v}^{\intercal}$. The resulting tensor is a $\vec{v}_{size} \times \vec{v}_{size}$ matrix, but we only need to keep the information about its upper triangular coefficients and because of that, the size required to store this tensor is equals to

$$\mathbf{T}_{size} = \vec{f}_{size} \times \left(\frac{\vec{f}_{size} + 1}{2} \right).$$

While vectors keep the information of one direction, orientation tensors can keep the tendency of many directions. In the Figure 3.13 we can see the different shapes that a 3-dimensional second-order tensor can have. The same idea is extended to higher dimensions.

The orientation tensor has two interesting properties explored in this work. The first one is the power to keep information about the relationship of the original vector's coefficients. This might come in handy when we want to combine different sources of information. For example, our descriptor is the result of the concatenation between the histogram of cross products with the coordinates in space and time of that trajectory. So, our tensor is capable of, not only encode information from the original sources but also

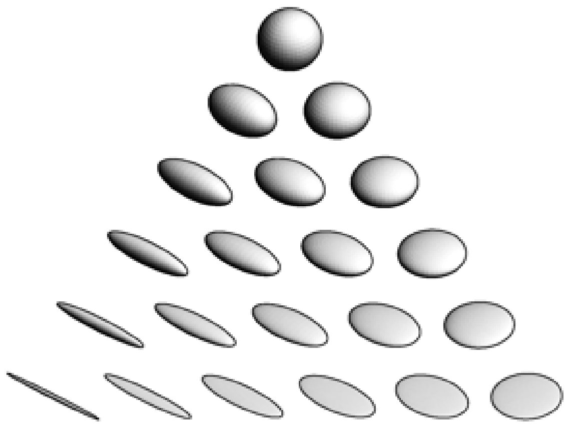


Figure 3.13: An illustration of different tensors (KINDLMANN, 2004).

their relationship. In other words, if there is a correlation between that information, the tensor will highlight it. The second property about orientation tensors explored in this work is that they are capable of accumulating information when summed up. So, when we sum two different tensors, the resulting tensor accumulates information about both, without increasing its dimension. But there are limitations, as pointed out by Mota et al. (2013), adding up too much information in a single tensor could lead it to a shape where there is no dominant direction — an isotropic tensor.

4 PROPOSED METHOD

Our proposed method has two main contributions. The first one is the cross product descriptor, which is based on the histogram of the cross product between the local trajectory displacement vector and the gradients in a surrounding window. The cross product descriptor is capable of encoding both motion and shape information in a single representation. The second main contribution is a strategy to cluster trajectories based on their shape. We create a vector for each trajectory that represents its shape regardless its direction. By applying k-means to cluster those vectors, we aggregate trajectories that represents the same type of movement in the same cluster.

In this work we use the dense trajectory method proposed by Wang et al. (2011) to extract trajectories. Note that our approach is not bounded to the dense trajectory method and can be implemented on top of any trajectory method. As the dense trajectories showed great results in many different works (JIANG et al., 2012; WANG et al., 2012; JAIN et al., 2013; WANG; SCHMID, 2013), we chose it as our trajectory extractor.

4.1 THE CROSS PRODUCT DESCRIPTOR

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

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

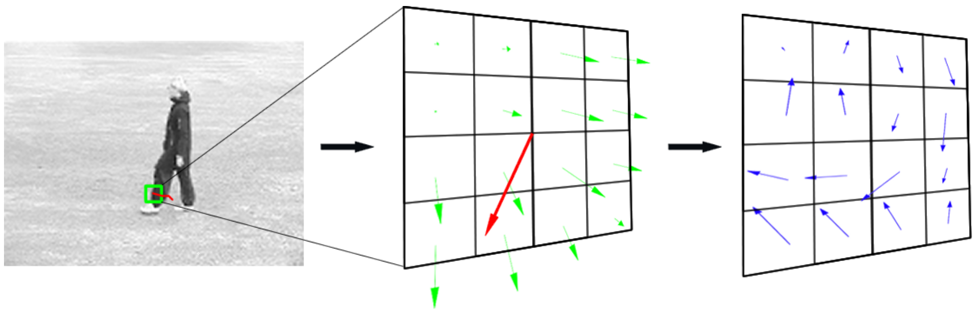


Figure 4.1: Overview of the cross product descriptor calculation. The picture in the left represents the frame with a trajectory tracked in a point. The picture in the middle represents the 3D gradient field (green vectors) in a surrounding window around that point and one displacement vector of the trajectory (red vector). The picture in the right is a representation of how the final vector field looks after the cross product between the trajectory displacement vector and each one of the gradients is computed.

time, which is one frame, making it $(u, v, 1)$.

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

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

where $\mathbf{q}$ is a point in the window $M_{\mathbf{p}_{j,i}}$ around the point $\mathbf{p}_{j,i}$ and $\vec{g}_{\mathbf{q}}$ is the 3D gradient in the point $\mathbf{q}$. An overview is presented in the Figure 4.1.

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

The resulting vector $C_{j,i}$ field is then quantized into a 3D histogram using a spherical coordinate system, the same way it is done in the HOG3D approach presented in the Section 3.1. We denote $\vec{h}_{i,j}$ as a vector corresponding to the histogram of the cross product of the trajectory i in its j-th displacement vector.

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

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

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

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

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

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

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

4.2 CLUSTERING TRAJECTORIES BY SHAPE

Dealing with camera motion is one of the most frequent problems in the human action recognition field. To address this problem, we propose a method to cluster trajectories based on their shape. Our hypothesis is that clustering trajectories by shape makes it possible to separate trajectories that are created by the camera's motion and that they are characterized by having the highest angle between each connected local displacement vector. Those trajectories are not discarded, as they are still source of information but, separated from the rest. Additionally we also expect that, even when there is no camera movement, separating trajectories with different shapes might improve the distinguishing power of our descriptor.

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

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

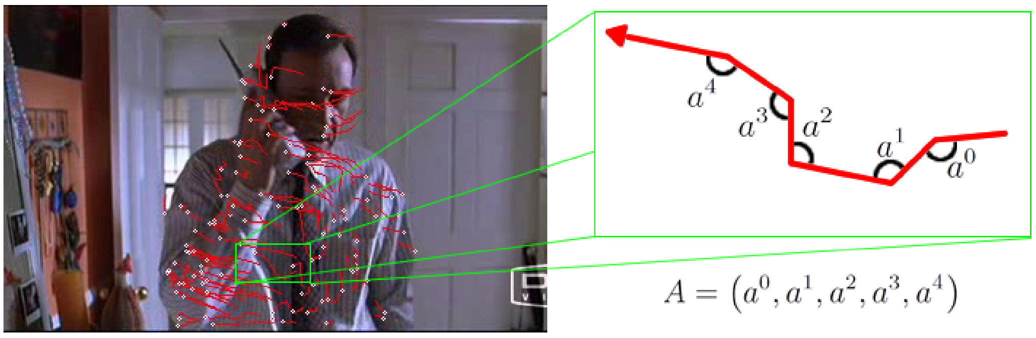


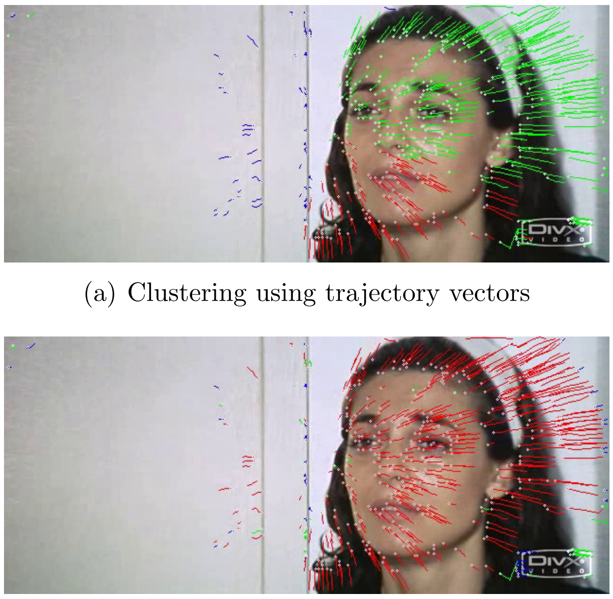
Figure 4.2: Illustration of how the vector of angles is computed.

where

$$a_{j,i} = \cos^{-1}\left(\frac{\vec{v}_{j,i} \cdot \vec{v}_{j+1,i}}{\|\vec{v}_{j,i}\| \cdot \|\vec{v}_{j+1,i}\|}\right). \tag{4.1}$$

The Equation 4.1 gives us the smallest angle between two connected displacement vectors in the trajectory. The resulting number is between 0 and 180 and the higher the number is, straighter is that connection. The Figure 4.2 shows an example of how the vector is computed for a specific trajectory. After that, we use k-means to cluster the trajectories using each of its angles vector. The mean of all angles of each cluster is then used to sort the cluster and determine how straight that group is. This step is required because the k-means does not give us any information about the clusters, just a symbolic label. Classifying the clusters using this metric increase the chance that most of the camera movement will rely on the first cluster as it tends to be straighter than trajectories generated by other movements. Also, trajectories are clustered in the same range in time. In other words, trajectories that have started at the frame t are clustered together. This enforces the need to sort the clusters to match them between trajectories that have started in different frames.

The idea of using the angles to cluster trajectories comes from the fact that clustering only the original trajectories vectors cannot deal with trajectories created by the zoom of the camera. An illustration of this effect can be seen in the Figure 4.3. In 4.3a trajectories created by the zoom of the camera are clustered mostly by directions, and not by shape. But as those trajectories do not have the same direction, they have a tendency to be clustered in different clusters. In 4.3b, we can see that using the vector of angles, trajectories are clustered by their actual shape, and not by their directions and thus, most of the trajectories generated by the camera movement are clustered together.



(a) Clustering using trajectory vectors

(b) Clustering using vector of angles

Figure 4.3: An illustration of two different approaches to cluster trajectories. The trajectories are clustered in three clusters and the color of the trajectory indicates which cluster it is associated with. Red indicates the cluster with higher mean angle, blue and green indicates lower mean angles.

4.3 CALCULATING THE FINAL DESCRIPTOR

In order to generate the final descriptor, we use both cross product descriptor and shape-based clustering approaches. An overview of our method is illustrated in the Figure 4.4.

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

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

where $\mathbf{R}_b^t$ is the tensor of the cluster $b \in [0, k-1]$ in the frame t, and $\boldsymbol{g}$ is a function that determines to which cluster a trajectory i belongs, after the trajectories are already clustered by k-means. These tensors are then normalized using the L^2-norm to keep its directions discarding the magnitude. This step is required to ensure that clusters with

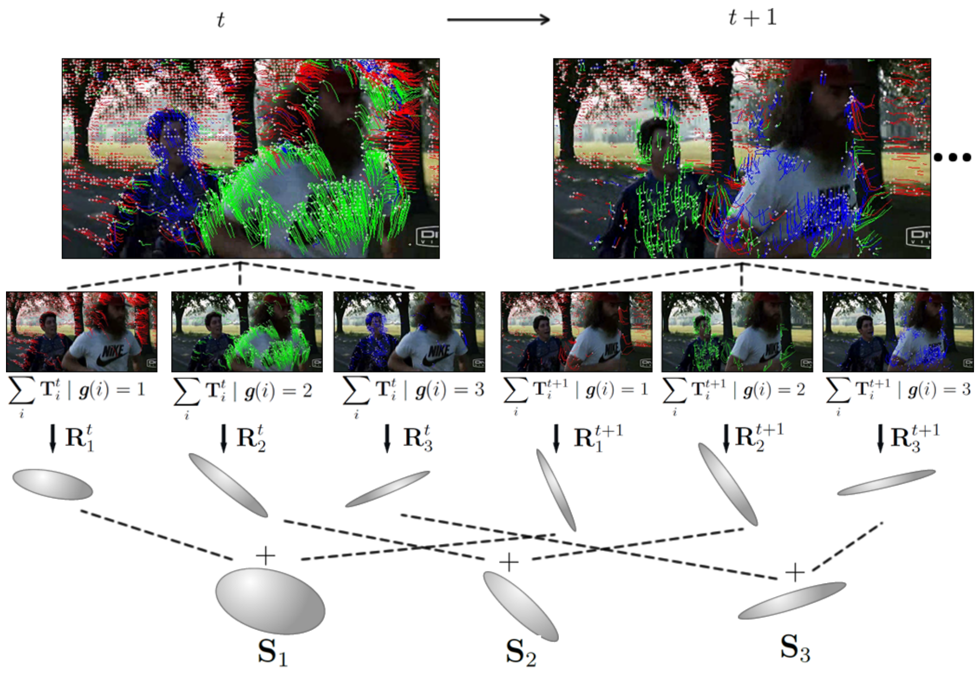

t
$t+1$
$\sum_i \mathbf{T}_i^t \mid g(i) = 1$
$\sum_i \mathbf{T}_i^t \mid g(i) = 2$
$\sum_i \mathbf{T}_i^t \mid g(i) = 3$
$\sum_i \mathbf{T}_i^{t+1} \mid g(i) = 1$
$\sum_i \mathbf{T}_i^{t+1} \mid g(i) = 2$
$\sum_i \mathbf{T}_i^{t+1} \mid g(i) = 3$
$\mathbf{R}_1^t$
$\mathbf{R}_2^t$
$\mathbf{R}_3^t$
$\mathbf{R}_1^{t+1}$
$\mathbf{R}_2^{t+1}$
$\mathbf{R}_3^{t+1}$
+
+
+
$\mathbf{S}_1$
$\mathbf{S}_2$
$\mathbf{S}_3$

Figure 4.4: Overview of our method. Trajectories that start in the frame t are clustered by their shape into k clusters. The figure shows an example for $k = 3$ clusters.

higher population will not generate a bigger tensor, compared to other clusters. After that, we have k tensors for each frame in the video. Since clusters are ordered using a metric, we can match tensors from different frames together. Thus, all tensors from the same cluster b from different frames are summed using

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

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

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

5 EXPERIMENTAL RESULTS

To evaluate the applicability of our descriptor in the human action recognition problem, we chose two of the most used datasets in the area: KTH and Hollywood2. All clustering steps are done using a sequential k-means[1] implementation with the stopping criterion of 500 iterations or less than 0.01% of change in the cluster membership from last iteration. We used the default dense trajectory method[2] proposed by Wang et al. (2011) to extract trajectories. Our data is classified using a non-linear support vector machine (CORTES; VAPNIK, 1995), using Gaussian and Triangle kernel and an one-against-all strategy for multiclass classification.

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

5.1 DATASETS

5.1.1 KTH

This database was proposed by Schuldt et al. (2004) and is composed of 6 actions: walking, jogging, running, boxing, hand waving and hand clapping. These actions are performed several times by 25 people in four different scenarios: outdoors, outdoors with camera zoom, outdoors with changed clothes and indoors. All sequences are taken with homogeneous backgrounds and a static camera. The KTH dataset has a total of 2391 sequences, divided into a training set (8 people), a validation set (8 people) and a test set (9 people). The classifier is trained with the training set, the validation set is used to optimize the parameters of the classifier and the final recognition rate is based on the number of correctly predicted actions for the test set. The Figure 5.1 shows a sample of the videos from the dataset.

[1]Available at http://www.ece.northwestern.edu/~wkliao/Kmeans/index.html
[2]Available at http://lear.inrialpes.fr/people/wang/dense_trajectories

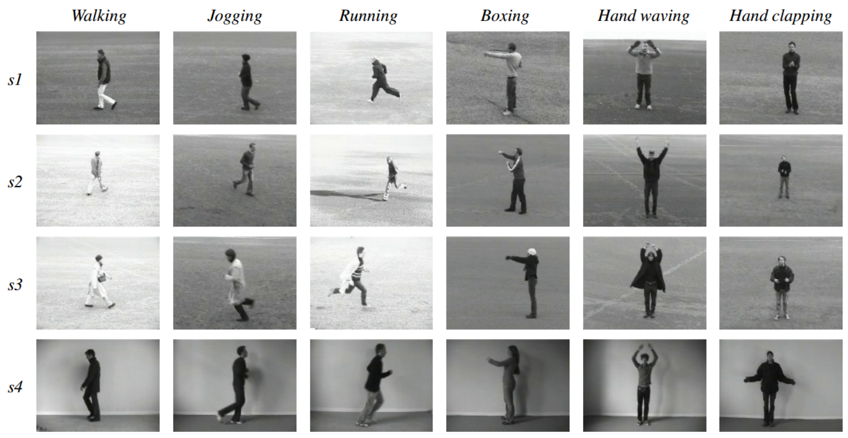


Figure 5.1: Illustration of KTH actions (SCHULDT et al., 2004).

Figure 5.2: Illustration of the Hollywood2 actions (MARSZALEK et al., 2009).

5.1.2 HOLLYWOOD 2

The Hollywood2 dataset was proposed by Marszalek et al. (2009) and is composed by 12 actions: answering phone, driving car, eating, fighting, getting out of car, hand shaking, hugging, kissing, running, sitting down, sitting up, and standing up. The sequences are made by video clips extracted from 69 different Hollywood movies and many challenges like camera movement, illumination conditions and background clutter are applied. There are a total of 1707 sequences in the dataset, which are divided in a training set with 823 samples and a test set with 884 samples. The classifier is trained with the training set and the final recognition performance is measured by the mean average precision (mAP), which is the mean of the percentage of correctly predicted actions of the test set for each class. An example of the sequences is shown in the Figure 5.2.

5.2 PARAMETERS

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

- Number of clusters: k represents the number of clusters the trajectories are grouped to. When $k = 1$, it means that our approach to cluster trajectories is not applied and thus a single tensor for all trajectories is computed for the whole frame and consequently for the entire video. There is a trade-off between increasing and decreasing the k. Low values for k may lead to different movement types grouped in the same cluster. Higher values for k, not just increase linearly the size of the final descriptor but there also is a chance to cluster similar movements in different clusters;

- Histogram bins: bin represents the number of subdivisions in the histogram of cross product. Low values to bin do not acquire the information we need and lead to a histogram that is not able to capture the cross product field direction's tendency. High values to bin do not summarize well the information and does not match well cross product fields that are similar but slightly different. Also, increasing the bin increases quadratically the size of the final descriptor so, higher numbers should be avoided;

- Window size: $M \times M$ is the size of the window around each point of the trajectory where the cross product is calculated. A small window size may not capture important features along the trajectory and a big window size may overlap between nearby trajectories and acquire redundant or too much information that should be discarded. Due to the complexity of our parameter space, we fixed the window size to a default value of 16×16, based on preliminary experiments. We are going to analyze the impact of changing the window size using our best result in subsections 5.3.1.6 and 5.3.2.5;

- Sampling stride: w defines the space between points that compose the sampling grid for new trajectory points. In other words, points that are candidate to start a trajectory are sampled in a grid spaced by w pixels in both x and y dimensions. A lower number for stride increases the coverage of the method, because more points are being sampled. The downside is that more trajectories are extracted and more computation power is required to process that information. Also, sampling lots of points does not necessary increase the amount of meaningful information, as the trajectories windows tend to overlap. Unless otherwise stated, the value for the stride is 5, as it is done in the original dense trajectory work (WANG et al., 2011);

- Cornerness of the point: the parameter q of the dense trajectory defines the minimum value for the maximum eigenvalue of a point so it can start a trajectory. A higher q increases the pressure to the candidate points, so the cornerness measure, defined by the Harris detector should be higher and more corner-like points are selected. In opposition to that, a lower q allows more points to be tracked but increases the chance of getting points that rely in flat or edge regions and are less reliable. Just like the dense trajectory work, our default value for q is 0.001;

- Maximum scale depth: dense trajectories are extracted in multiple scales individually to achieve some scale invariance. Increasing the maximum scale grants a better coverage of the method but also increases its computational complexity. The default value for the maximum scale depth in our methods is 5. The factor of each scale is $f =^1 /_{\sqrt{2}}$, which means that each new scale has half the area of the previous scale;

- Trajectory size: l defines for how many frames the initial point will be tracked to compose the trajectory. Low values for l will generate short trajectories that does not encode properly the executed motion. High values for the trajectory size lead to longer trajectories that are more likely to drift from the original tracked point and therefore are less reliable;

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

- SVM kernel: we use both Triangle and Gaussian kernel on the SVM classifier in our experiments.

5.3 EXPERIMENTS

5.3.1 KTH DATASET

For the KTH dataset we tested the following range of parameters: $l \in \{5, 10, 20\}$, $bin \in \{(6 \times 3), (10 \times 5), (16 \times 8), (18 \times 9), (22 \times 11), (24 \times 12)\}$, $k \in \{1, 2, 3, 5, 10\}$, $w \in \{2, 5\}$ using the factorial experiment, which means that every permutation of those parameters is considered. To measure the impact of each parameter, we calculate the mean, standard deviation, minimum and maximum recognition rate for each one of them. These values are computed using the results from the permutation of all parameters, but individually fixing each value of analyzed parameter.

5.3.1.1 Number of clusters

To analyze the impact of the number of clusters in our method, we computed the mean recognition rate in the KTH dataset for each value of k. The result can be seen in the Table 5.1. One may see that not applying the proposed clustering strategy for the trajectories (i.e. $k = 1$) decrease the average recognition rate by a reasonable amount. In average, the best configuration was for $k = 5$.

Table 5.1: Recognition rate analysis for each number of clusters used to group trajectories in the KTH dataset. Each value of k was tested 432 times with different parameters.

k	Min	Max	Avg	SD
1	87.10%	93.50%	91.56%	1.32%
2	88.00%	93.70%	91.85%	0.99%
3	**88.90%**	93.90%	92.03%	0.86%
5	87.50%	**94.10%**	**92.19%**	0.94%
10	87.80%	93.90%	92.17%	0.92%

It is interesting to notice that the KTH dataset does not contain much camera's movement. Even so, clustering the trajectories shows the best result in average.This is an indication that our clustering method is not just a good strategy to separate the trajectories made by the camera, but also to separate different movement types. Also, as the k goes higher, the number of information on each final tensor is smaller. In other words, the information that would be stored in just one tensor using $k = 1$ is distributed over more tensors for the video, as the value of k is increased. That might be a sign that using just one tensor for the whole video might end up with an isotropic tensor, where there is no dominant direction.

5.3.1.2 Number of bins

The Table 5.2 shows the comparison between each value of bin. We can see that by using even a very small 6×3 value for bin, there is still a fair mean recognition rate. This is an evidence that the cross product descriptor has a good discriminative power. Additionally, using $bin = 10 \times 5$ showed a very good result, comparable with the highest bins. This is an important feature because it shows that the cross product descriptor is suitable even when a smaller descriptor is necessary. For instance, the descriptor size for both 6×3 and 10×5 are respectively 231 and 1431 coefficients, using $k = 1$. We can see that there is a tendency to increase the mean recognition rate together with the number of bins and $bin = 24 \times 12$ showed the best results for the dataset.

Table 5.2: Recognition rate analysis for each number of histogram bins in the KTH dataset. Each value of bin was tested 360 times with different parameters.

bin	Min	Max	Avg	SD
6×3	87.10%	92.60%	90.35%	1.10%
10×5	89.60%	93.90%	92.12%	0.70%
16×8	90.40%	93.60%	92.27%	0.63%
18×9	90.00%	93.70%	92.32%	0.67%
22×11	90.00%	94.00%	92.23%	0.62%
24×12	**90.50%**	**94.10%**	**92.48%**	0.68%

5.3.1.3 Trajectory's size

The Table 5.3 shows the comparison between each value of l. The value $l = 10$ shows a slightly better result than $l = 5$ in average but it is not enough to assert which trajectory size would be ideal for our method. Our results for $l = 20$ showed that there is a tendency to decrease the recognition rate for long trajectories. This is expected because trajectories that are too long tends to drift from the their real locations.

Table 5.3: Recognition rate analysis for each trajectory's size in the KTH dataset. Each value of l was tested 720 times with different parameters.

l	Min	Max	Avg	SD
5	87.10%	93.70%	91.88%	1.11%
10	**88.20%**	**94.10%**	**92.13%**	0.95%
20	87.50%	93.90%	91.87%	1.05%

5.3.1.4 Stride

The Table 5.4 shows the comparison between both values of w. The default value for the stride is 5 but, we also tested a second value for w to analyze its response in the recognition rate. This step was necessary because the resolution for the KTH dataset videos is very small and thus, some videos had very few trajectories created. Reducing the gap between the sampled points is the strategy applied to increase the number of the trajectories. We can see that, although $w = 2$ really shows a better recognition rate in the average, it is not that significant — about 0.035. Thus, $w = 5$ would be a better option if faster computation is crucial, as a bigger stride means fewer points computed and less trajectories tracked.

Table 5.4: Recognition rate analysis for each sampling stride in the KTH dataset. Each value of w was tested 1080 times with different parameters.

w	Min	Max	Avg	SD
2	**87.50%**	**94.10%**	**91.98%**	1.01%
5	87.10%	93.90%	91.94%	1.08%

5.3.1.5 Power normalization and kernel method

The Table 5.5 shows the comparison between both values of the power normalization and the kernel for the SVM. We can clearly see that the triangle kernel presented a better result in the classification task than the Gaussian kernel. Also, 0.2, 0.25 and 0.27 showed to be the best value for the power normalization, among the chosen values.

Table 5.5: Recognition rate analysis in the KTH for each normalization coefficient and each kernel applied. Each value of α was tested 360 times with different parameters. Each kernel was tested 1080 times with different parameters.

α	Min	Max	Avg	SD
0.3	88.00%	94.00%	92.02%	1.05%
0.27	87.50%	**94.10%**	92.16%	1.03%
0.25	87.50%	93.90%	**92.20%**	1.00%
0.2	**88.10%**	93.90%	92.16%	0.97%
0.15	87.80%	93.70%	91.91%	0.99%
0.1	87.10%	93.10%	91.30%	0.96%

kernel	Min	Max	Avg	SD
triangle	**88.30%**	**94.10%**	**92.17%**	0.96%
gauss	87.10%	94.00%	91.75%	1.09%

5.3.1.6 Analysis of the best result

The Table 5.6 shows our best result for the KTH dataset. The parameters are $k = 5$, $bin = 24 \times 12$, $l = 10$, $w = 2$, $\alpha = 0.27$ using a triangle kernel. The overall recognition rate is 94.1%. In the confusion matrix we can see the classes Boxing, Hand Clapping, Hand Waving and Walking achieved a very good recognition rate. Most of the wrong predictions are concentrated mutually between Running and Jogging and between Walking and Jogging. This is expected because in the KTH dataset, those movements are very similar and, for as much as we want the descriptor to be able to complete differentiate them, sometimes there is an intersection in the actions performed that are just impossible to separate.

Table 5.6: Confusion matrix for our best result for the KTH dataset. The recognition rate is 94.1%.

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

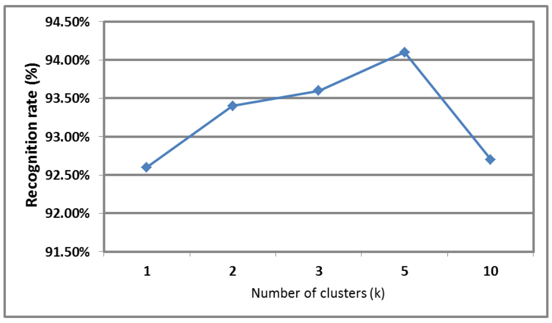
In the Figure 5.3 we analyze the impact of modifying each one of the parameter in our best result. In every case, the default values for our best configuration are kept, changing only the parameter we are interested in. In the Figure 5.3a we can see that increasing k improved the recognition rate until $k = 5$ and dropped for $k = 10$. In the Figure 5.3b we vary the values for bin. In general, there is a tendency to improve the recognition rate as bin increases but, even for small values our method is still has a fair performance. The trajectory size l is evaluated in the Figure 5.3c and $l = 10$ showed as the best result, approximately 0.5% higher recognition rate than $l = 5$ and $l = 20$. In the Figure 5.3d we can see the impact of the varying the stride. For the best result, changing w showed a great difference. The value $w = 2$ is about 1% higher in the recognition rate than $w = 5$. We can see in the Figure 5.3e that the kernel chosen to classify the data had a very significant impact in our best result. The triangle kernel showed an approximately 1.5% greater result than the Gaussian kernel. The Figure 5.3f shows the impact of changing the power normalization coefficient. $\alpha = 0.27$ showed the best result overall but $\alpha = 0.25$ has a comparable recognition rate. For last, the Figure 5.3g shows that changing the window size from its default value does not lead to great changes in our method's recognition rate.

5.3.2 HOLLYWOOD2 DATASET

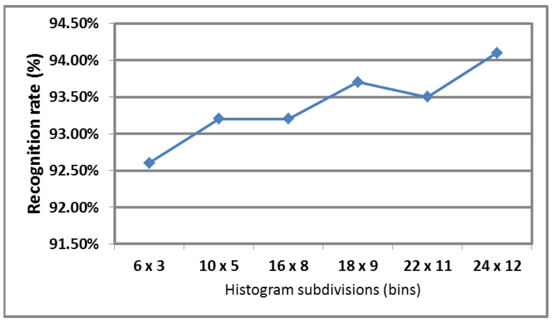
For the Hollywood2 dataset we tested the following range of parameters: $l \in \{5, 10, 15\}$, $bin \in \{(6 \times 3), (10 \times 5), (16 \times 8), (22 \times 11), (24 \times 12), (26 \times 13)\}$, $k \in \{1, 2, 3, 5, 10\}$ using the factorial experiment, which means that every permutation of those parameters is considered. Just like the experiments in the KTH dataset, we measure the impact of each parameter, fixing each one of its values and varying all the other parameters.

5.3.2.1 Number of clusters

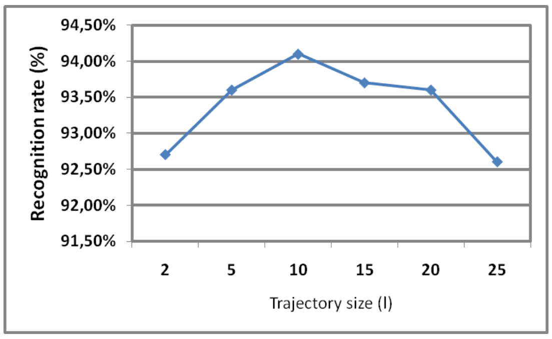
The Table 5.7 shows the impact of changing the number of clusters in the Hollywood2 dataset. Analyzing the average values for each value of k, we can presume that increasing the number of clusters has a positive impact in the recognition rate. But looking at the maximum values for each parameter, we can see that all values for k except $k = 1$ reached a good recognition rate in the database. The value for k with the best mean average precision is 10. Although increasing the number of clusters shows a very good result, one should analyze whether it is feasible or not. Increasing the number of clusters linearly increases the final descriptor size. This is also why we did not test any further values for k.



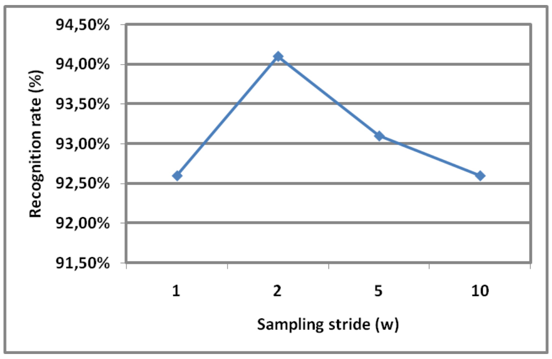
(a)



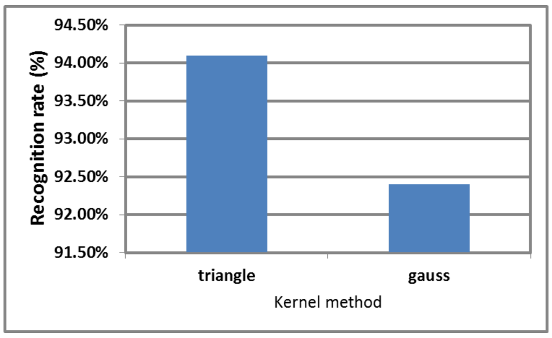
(b)



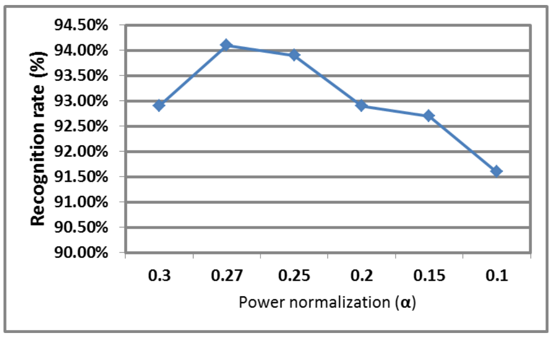
(c)



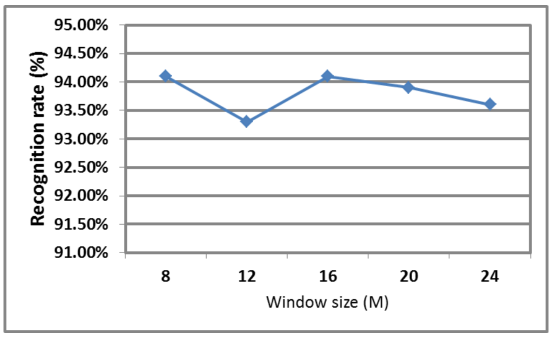
(d)



(e)



(f)



(g)

Figure 5.3: Impact of parameters variation in the best result for the KTH dataset.

Table 5.7: Recognition rate analysis in the Hollywood2 dataset for each number of clusters used to group trajectories. Each value of k was tested 216 times with different parameters.

k	Min	Max	Avg	SD
1	26.70%	45.00%	39.54%	3.60%
2	29.00%	46.10%	40.92%	3.23%
3	29.30%	46.10%	41.20%	3.20%
5	**30.10**%	**46.30**%	41.43%	3.12%
10	29.90%	46.10%	**41.63**%	3.25%

5.3.2.2 Number of bins

We measure the impact of different bins on the Hollywood2 dataset in the Table 5.8. Just like the KTH dataset, we can see that increasing the number of subdivisions in the histogram have a positive effect in the recognition rate. In average, the value for bin that showed the best result is 26×13. Increasing it too much would lead to a final descriptor that is too big to the required application. For example, each tensor using a histogram with $bin = 26 \times 13$ plus its spatio-temporal information (x, y, t) has 58311 coefficients. While the same tensor, using $bin = 10 \times 5$ has only 1431 coefficients.

Table 5.8: Recognition rate analysis for each number of subdivisions in the histogram in the Hollywood2 dataset. Each value of bin was tested 180 times with different parameters.

bin	Min	Max	Avg	SD
6×3	26.70%	42.20%	35.33%	3.20%
10×5	35.00%	45.40%	40.18%	2.07%
16×8	37.50%	45.60%	41.68%	1.72%
22×11	**39.10**%	46.10%	42.64%	1.48%
24×12	38.80%	46.00%	42.73%	1.56%
26×13	38.70%	**46.30**%	**43.11**%	1.53%

5.3.2.3 Trajectory's size

The trajectory size showed as a very important parameter in our method for the Hollywood2 dataset. The Table 5.9 shows the mean average precision for each fixed trajectory's size. The value $l = 5$ clearly shows a better performance over other values.

Table 5.9: Recognition rate analysis for each trajectory's size in the Hollywood2 dataset. Each value of l was tested 360 times with different parameters.

l	Min	Max	Avg	SD
5	26.70%	**46.30%**	**41.32%**	**3.52%**
10	27.50%	45.30%	40.62%	3.30%
15	**27.70%**	45.40%	40.89%	3.23%

5.3.2.4 Power normalization and kernel method

The impact of the power normalization and the kernel method on the Hollywood2 dataset can be seen in the Table 5.10. We can see that $\alpha = 0.20$ together and the triangle kernel shows the best mean recognition rate in the database. As using the power normalization in the descriptor with $\alpha = 0.2$ is the same of taking the 5-th root of each coefficient, having such low values for α shows that our final descriptor is probably carrying peaks on it.

Table 5.10: Mean recognition rate in the Hollywood2 for each normalization coefficient and each kernel applied. Each value of α was tested 180 times with different parameters. Each kernel was tested 540 times with different parameters.

α	Min	Max	Avg	SD
0.3	26.70%	43.50%	39.07%	3.82%
0.27	28.20%	45.00%	40.45%	3.80%
0.25	29.60%	45.40%	41.22%	3.74%
0.2	32.90%	**46.30%**	**42.09%**	3.13%
0.15	**34.90%**	45.50%	41.85%	2.32%
0.1	**34.90%**	44.90%	40.98%	1.98%
kernel	Min	Max	Avg	SD
triangle	**41.37%**	26.70%	**46.30%**	3.92%
gauss	40.51%	**31.60%**	45.40%	2.63%

5.3.2.5 Analysis of the best result

The Table 5.11 shows the best recognition rate achieved by our method in the Hollywood2 dataset. The results were obtained by using $k = 5$, $bin = 26 \times 13$, $l = 5$, $w = 5$, $\alpha = 0.20$ and the triangle kernel.

In the Figure 5.4 we can see the impact of changing the parameters for our best result. Changing the value for k in the Figure 5.4a shows that when $k = 1$ and our strategy

Table 5.11: Average precision for each class of Hollywood2 for our best configuration. The mean average precision is 46.3%.

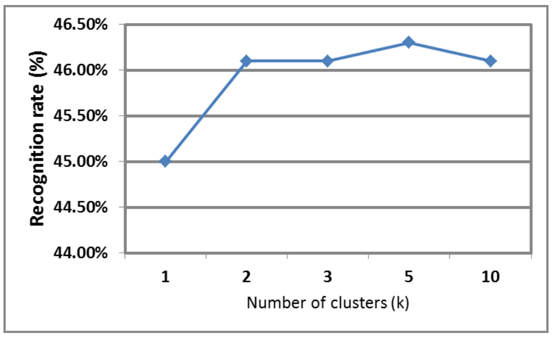
Action	APhone	DCar	Eat	FPerson	GetOutCar	HShake
AP(%)	26.5	85.2	59	58	29.8	32.3
Action	HPerson	Kiss	Run	SDown	SUp	StandUp
AP(%)	24.3	48.3	66.1	54.3	19.6	52.8

to cluster trajectories is not applied the mean recognition rate decreases for a reasonable amount. The best result was achieved with $k = 5$ but, other values for k still reaches a good performance. The Figure 5.4b shows that increasing the number of bins in the histogram has a positive effect in the final recognition rate, although it tends to stabilize after $bin = 22 \times 11$. The trajectory size is evaluated in the Figure 5.4c and $l = 5$ shows the best result over $l = 5$ and $l = 15$. We can see in the Figure 5.4d that the chosen kernel also had a very significant impact in our best result. The triangle kernel showed an approximately 1.25% greater result than the Gaussian kernel. The Figure 5.3e shows the impact of changing the power normalization coefficient. $\alpha = 0.2$ showed the best result overall. The Figure 5.4f shows that the recognition rate in function of the window size for the Hollywood2 dataset. Note that using a window of 12×12 improved the best result by 0.2%, which is a significant improvement. Unfortunately, we could not perform experiments with the same parameters used with the 16×16 due to time limitations.

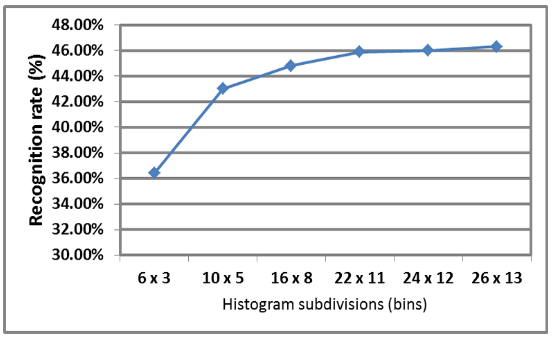
5.4 COMPARISON WITH PREVIOUS WORKS

In this section we compare our best results with the state-of-the-art in literature. To compare our results, we need to take into account that most of the methods are not restricted to the self-descriptor constraint and use approaches similar to the BoF. That means that their final descriptor for each video is calculated using the information of all videos merged together. In the Table 5.12 we present a comparison between our best results and these works.

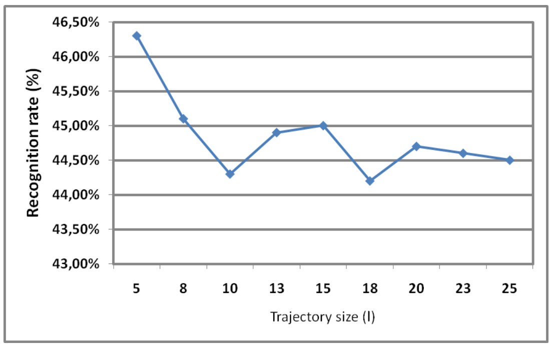
We can see that, among self-descriptor works, our method shows the best performance for both KTH and Hollywood2 datasets. For the KTH, we achieve a recognition 0.8% better than the previous best result for self-descriptor methods and a comparable result against dictionary-based methods. For the Hollywood2, we improved the best result for



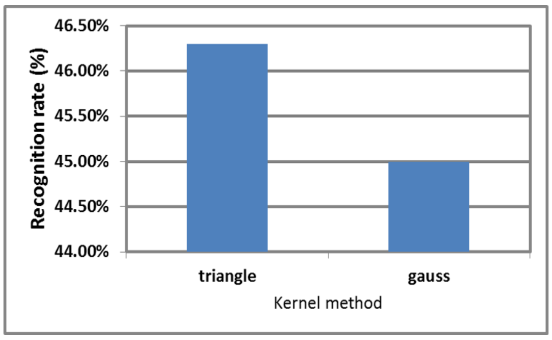
(a)



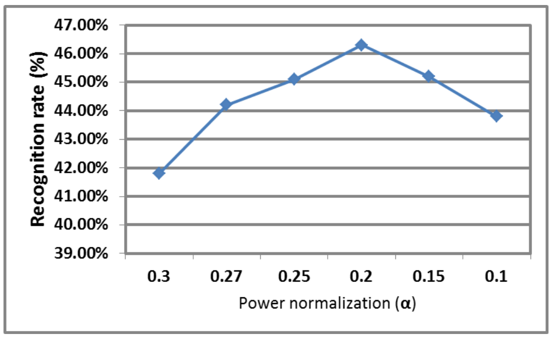
(b)



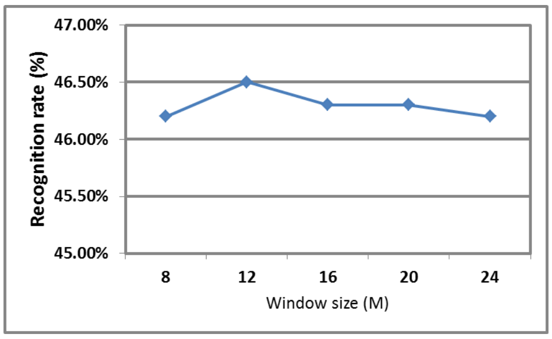
(c)



(d)



(e)



(f)

Figure 5.4: Impact of parameters variation in the best result for the Hollywood2 dataset.

Table 5.12: Comparison with state-of-the-art for KTH and Hollywood2 datasets. The best results of our method were obtained with 16×16 and 12×12 window size for KTH and Hollywood2, respectively.

	KTH	**Hollywood2**
Dictionary-based Methods		
Wang et al. (2011)	94.2	58.2
Kobayashi and Otsu (2012)	**95.6**	47.7
Wang et al. (2012)	95.3	59.9
Jain et al. (2013)		62.5
Wang and Schmid (2013)		**64.3**
Self-descriptor Methods		
Perez et al. (2012)	92.0	34.0
Mota et al. (2013)	92.5	40.3
Sad et al. (2013)	93.3	41.9
Mota et al. (2014)	93.2	40.3
Figueiredo et al. (2014)	87.7	34.9
Our method	**94.1**	**46.5**

self-descriptor methods by 4.4%. We believe this great improvement comes from the fact that the Hollywood2 is a challenging dataset as it contains lots of camera movement and many different scenarios. In this particular situation, our method to cluster trajectories can separate those movements quite well, which is not done in the previous self-descriptor works. The same impact can not be seen in the KTH dataset because it does not contain much camera movement and the background is very homogeneous. In despite of the great improvement for the Hollywood2, we can see that our method is still not able to reach the current state-of-start methods that are based on a dictionary of visual words (BoF, VLAD, Fisher Vector, etc.).

6 CONCLUSION

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

Our results showed that for the Hollywood dataset, the best setup can achieve 46.3% in the mean average precision, using 10 trajectory's clusters, trajectories with 5 frames long and 26×13 subdivisions in the histogram of cross product. For the KTH dataset, the best setup can achieve 94.1% using 5 trajectory's clusters, trajectories with 10 frames long, 24×12 subdivisions in the histogram of cross product and sampling trajectories in a 2 pixel spaced grid.

Due to the wideness of our parameter space, the window size parameter M was initially fixed on 16×16. We used the best result found with this configuration to analyze the impact of changing the window size for both databases. Our results showed that other values for M does not have a strong impact the final result. At the last moment of our work, we performed tests with the window size 12×12 in the Hollywood2 dataset, achieving 46.5%. Due to the dissertation's deadline we could not compute the recognition rates varying parameters as we did with the 16×16 window. This result shows that further experiments are necessary to explore the parameter space. However, we believe that the experiments using 12×12 will not be significantly higher than the exposed in this work.

One suggestion to future works is to cluster all trajectories from the whole video just one time. This is different to the strategy to cluster trajectories applied in this work. In our method we cluster trajectories that start in the same frame. This is a good idea because our strategy to match clusters from different frames using the mean of its angles may not always work. For example, consecutive frames with very different motion patterns will end up to be summed in the same tensor. The problem with clustering all video's trajectories just one time is that it would require a bigger space in the main memory. The descriptors are extracted along trajectories and it would be necessary to store them before clustering. Thus, to address this problem a new data structure and clustering method using the secondary memory should be taken into account.

An important investigation that can be done in the future is to analyze different strategies to quantize the 3D histogram. Currently, we quantize θ and ϕ to equally spaced bins, which is known for having a singularity on the poles showed in the Figure 3.2. We apply this strategy here because, when quantizing the HOG3D, previous works showed no difference between this and the platonic solids approach proposed by Klaser et al. (2008). This behavior may not occur with the cross product descriptor due to a different distribution over the bins. For example, the cross product could have more vectors matching in the bins located at the poles than the HOG3D, therefore being more affected by the known singularity in those areas. Further analysis can show the impact of changing the quantize method for the cross product descriptor.

In our results we noticed that very low values for the power normalization showed to be the best results. This might be an effect of dealing with tensors in the Euclidean space, instead of the Log-Euclidean space. We tried to apply the Log-Euclidean in our tensors as done by Yuan et al. (2010). But we had numerical precision problems on log and exponential functions on higher numbers. Thus, further studies using Log-Euclidean are recommended.

REFERENCES

BAEZA-YATES, R. A.; RIBEIRO-NETO, B. **Modern Information Retrieval**, 1999. ISBN 020139829X.

BAKER, S.; SCHARSTEIN, D.; LEWIS, J. P.; ROTH, S.; BLACK, M. J.; SZELISKI, R. A database and evaluation methodology for optical flow. **Int. J. Comput. Vision**, Kluwer Academic Publishers, Hingham, MA, USA, v. 92, n. 1, p. 1–31, mar. 2011. ISSN 0920-5691. Disponível em: <http://dx.doi.org/10.1007/s11263-010-0390-2>.

BAY, H.; ESS, A.; TUYTELAARS, T.; GOOL, L. V. Speeded-up robust features (surf). **Comput. Vis. Image Underst.**, Elsevier Science Inc., New York, NY, USA, v. 110, n. 3, p. 346–359, jun. 2008. ISSN 1077-3142. Disponível em: <http://dx.doi.org/10.1016/j.cviu.2007.09.014>.

BLANK, M.; GORELICK, L.; SHECHTMAN, E.; IRANI, M.; BASRI, R. Actions as space-time shapes. In: **Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on**, 2005. v. 2, p. 1395–1402 Vol. 2. ISSN 1550-5499.

CORTES, C.; VAPNIK, V. Support-vector networks. **Mach. Learn.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 20, n. 3, p. 273–297, set. 1995. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1022627411411>.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: **Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on**, 2005. v. 1, p. 886–893 vol. 1. ISSN 1063-6919.

DALAL, N.; TRIGGS, B.; SCHMID, C. Human detection using oriented histograms of flow and appearance. In: **Proceedings of the 9th European Conference on Computer Vision - Volume Part II**, 2006. (ECCV'06), p. 428–441. ISBN 3-540-33834-9, 978-3-540-33834-5.

DOLLAR, P.; RABAUD, V.; COTTRELL, G.; BELONGIE, S. Behavior recognition via sparse spatio-temporal features. In: **Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on**, 2005. p. 65–72.

FARNEBÄCK, G. Two-frame motion estimation based on polynomial expansion. In: **Proceedings of the 13th Scandinavian Conference on Image Analysis**, 2003. (SCIA'03), p. 363–370. ISBN 3-540-40601-8. Disponível em: <http://dl.acm.org/citation.cfm?id=1763974.1764031>.

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: MURGANTE, B.; MISRA, S.; ROCHA, A. M. A.; TORRE, C.; ROCHA, J. G.; FALCÃO, M. I.; TANIAR, D.; APDUHAN, B. O.; GERVASI, O. (Ed.). **Computational Science and Its Applications ICCSA 2014**, 2014, (Lecture Notes in Computer Science, v. 8584). p. 401–414. ISBN 978-3-319-09152-5. Disponível em: <http://dx.doi.org/10.1007/978-3-319-09153-2_30>.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM**, ACM, New York, NY, USA, v. 24, n. 6, p. 381–395, jun. 1981. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/358669.358692>.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: **In Proc. of Fourth Alvey Vision Conference**, 1988. p. 147–151.

JAIN, M.; JÉGOU, H.; BOUTHEMY, P. Better exploiting motion for better action recognition. In: **CVPR - International Conference on Computer Vision and Pattern Recognition**, 2013. Disponível em: <http://hal.inria.fr/hal-00813014>.

JEGOU, H.; DOUZE, M.; SCHMID, C.; PEREZ, P. Aggregating local descriptors into a compact image representation. In: **Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on**, 2010. p. 3304–3311. ISSN 1063-6919.

JIANG, Y.-G.; DAI, Q.; XUE, X.; LIU, W.; NGO, C.-W. Trajectory-based modeling of human actions with motion reference points. In: **Proceedings of the 12th European Conference on Computer Vision - Volume Part V**, 2012. (ECCV'12), p. 425–438. ISBN 978-3-642-33714-7.

KINDLMANN, G. L. Superquadric tensor glyphs. In: DEUSSEN, O.; HANSEN, C. D.; KEIM, D. A.; SAUPE, D. (Ed.). **VisSym**, 2004. p. 147–154. ISBN 3-905673-07-X. Disponível em: <http://dblp.uni-trier.de/db/conf/vissym/vissym2004.html#Kindlmann04>.

KLASER, A.; MARSZALEK, M.; SCHMID, C. A Spatio-Temporal Descriptor Based on 3D-Gradients. In: EVERINGHAM, M.; NEEDHAM, C.; FRAILE, R. (Ed.). **BMVC 2008 - 19th British Machine Vision Conference**, 2008. p. 275:1–10. Disponível em: <http://hal.inria.fr/inria-00514853>.

KOBAYASHI, T.; OTSU, N. Motion recognition using local auto-correlation of space-time gradients. **Pattern Recogn. Lett.**, Elsevier Science Inc., New York, NY, USA, v. 33, n. 9, p. 1188–1195, jul. 2012. ISSN 0167-8655. Disponível em: <http://dx.doi.org/10.1016/j.patrec.2012.01.007>.

KUEHNE, H.; JHUANG, H.; GARROTE, E.; POGGIO, T.; SERRE, T. Hmdb: A large video database for human motion recognition. In: **Computer Vision (ICCV), 2011 IEEE International Conference on**, 2011. p. 2556–2563. ISSN 1550-5499.

LAPTEV, I. On space-time interest points. **Int. J. Comput. Vision**, Kluwer Academic Publishers, Hingham, MA, USA, v. 64, n. 2-3, p. 107–123, set. 2005. ISSN 0920-5691. Disponível em: <http://dx.doi.org/10.1007/s11263-005-1838-7>.

LAPTEV, I.; MARSZALEK, M.; SCHMID, C.; ROZENFELD, B. Learning realistic human actions from movies. In: **Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on**, 2008. p. 1–8. ISSN 1063-6919.

LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: **Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2**, 1981. (IJCAI'81), p. 674–679. Disponível em: <http://dl.acm.org/citation.cfm?id=1623264.1623280>.

MARSZALEK, M.; LAPTEV, I.; SCHMID, C. Actions in context. In: **Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on**, 2009. p. 2929–2936. ISSN 1063-6919.

MOTA, V.; SOUZA, J.; ARAUJO, A. D. A.; VIEIRA, M. B. Combining orientation tensors for human action recognition. In: **Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI - Conference on**, 2013. p. 328–333. ISSN 1530-1834.

MOTA, V. F. **Tensor baseado em fluxo óptico para descrição global de movimento em vídeos**. Dissertação (Dissertação de Mestrado em Modelagem Computacional) — Universidade Federal de Juiz de Fora (UFJF), 2011.

MOTA, V. F.; PEREZ, E. A.; MACIEL, L. M.; VIEIRA, M. B.; GOSSELIN, P. H. A tensor motion descriptor based on histograms of gradients and optical flow. **Pattern Recogn. Lett.**, Elsevier Science Inc., New York, NY, USA, v. 39, p. 85–91, abr. 2014. ISSN 0167-8655. Disponível em: <http://dx.doi.org/10.1016/j.patrec.2013.08.008>.

MOTA, V. F.; PEREZ, E. A.; VIEIRA, M. B.; MACIEL, L. M.; PRECIOSO, F.; GOSSELIN, P. H. A tensor based on optical flow for global description of motion in videos. In: **Proceedings of the 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images**, 2012. (SIBGRAPI '12), p. 298–301. ISBN 978-0-7695-4829-6. Disponível em: <http://dx.doi.org/10.1109/SIBGRAPI.2012.48>.

PEREZ, E. de A.; MOTA, V. F.; MACIEL, L. M.; SAD, D. O.; VIEIRA, M. B. Combining gradient histograms using orientation tensors for human action recognition. In: **ICPR**, 2012. p. 3460–3463. ISBN 978-1-4673-2216-4. Disponível em: <http://dblp.uni-trier.de/db/conf/icpr/icpr2012.htmlPerezMMSV12>.

PERRONNIN, F.; SáNCHEZ, J.; MENSINK, T. Improving the fisher kernel for large-scale image classification. In: **Proceedings of the 11th European Conference on Computer Vision: Part IV**, 2010. (ECCV'10), p. 143–156. ISBN 3-642-15560-X, 978-3-642-15560-4. Disponível em: <http://dl.acm.org/citation.cfm?id=1888089.1888101>.

PICARD, D.; GOSSELIN, P.-H. Improving Image Similarity With Vectors of Locally Aggregated Tensors. In: **2011 IEEE International Conference on Image Processing (IEEE ICIP2011)**, 2011. p. 669 – 672. Disponível em: <http://hal.archives-ouvertes.fr/hal-00591993>.

PREST, A.; SCHMID, C.; FERRARI, V. Weakly supervised learning of interactions between humans and objects. **IEEE Trans. Pattern Anal. Mach. Intell.**, IEEE Computer Society, Washington, DC, USA, v. 34, n. 3, p. 601–614, mar. 2012. ISSN 0162-8828. Disponível em: <http://dx.doi.org/10.1109/TPAMI.2011.158>.

SAD, D.; MOTA, V. F.; MACIEL, L. M.; VIEIRA, M. B.; ARAÃºJO, A. d. A. A tensor motion descriptor based on multiple gradient estimators. In: **Proceed-**

ings of the 2013 XXVI Conference on Graphics, Patterns and Images, 2013. (SIBGRAPI '13), p. 70–74. ISBN 978-0-7695-5099-2. Disponível em: <http://dx.doi.org/10.1109/SIBGRAPI.2013.19>.

SCHULDT, C.; LAPTEV, I.; CAPUTO, B. Recognizing human actions: A local svm approach. In: **Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03**, 2004. (ICPR '04), p. 32–36. ISBN 0-7695-2128-2. Disponível em: <http://dx.doi.org/10.1109/ICPR.2004.747>.

SCOVANNER, P.; ALI, S.; SHAH, M. A 3-dimensional sift descriptor and its application to action recognition. In: **Proceedings of the 15th International Conference on Multimedia**, 2007. (MULTIMEDIA '07), p. 357–360. ISBN 978-1-59593-702-5. Disponível em: <http://doi.acm.org/10.1145/1291233.1291311>.

SIVIC, J.; ZISSERMAN, A. Video google: A text retrieval approach to object matching in videos. In: **Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2**, 2003. (ICCV '03), p. 1470–. ISBN 0-7695-1950-4. Disponível em: <http://dl.acm.org/citation.cfm?id=946247.946751>.

SUN, J.; MU, Y.; YAN, S.; CHEONG, L.-F. Activity recognition using dense long-duration trajectories. In: **Multimedia and Expo (ICME), 2010 IEEE International Conference on**, 2010. p. 322–327. ISSN 1945-7871.

SUN, J.; WU, X.; YAN, S.; CHEONG, L.-F.; CHUA, T.-S.; LI, J. Hierarchical spatio-temporal context modeling for action recognition. In: **Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on**, 2009. p. 2004–2011. ISSN 1063-6919.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 1st. ed., 2010. ISBN 1848829345, 9781848829343. Disponível em: <http://szeliski.org/Book>.

WANG, H.; KLASER, A.; SCHMID, C.; LIU, C.-L. Action recognition by dense trajectories. In: **Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on**, 2011. p. 3169–3176. ISSN 1063-6919.

WANG, H.; KLÄSER, A.; SCHMID, C.; LIU, C.-L. **Dense trajectories and motion boundary descriptors for action recognition**, ago. 2012. Disponível em: <http://hal.inria.fr/hal-00725627>.

WANG, H.; SCHMID, C. Action Recognition with Improved Trajectories. In: **ICCV 2013 - IEEE International Conference on Computer Vision**, 2013. p. 3551–3558. Disponível em: <http://hal.inria.fr/hal-00873267>.

WILLEMS, G.; TUYTELAARS, T.; GOOL, L. An efficient dense and scale-invariant spatio-temporal interest point detector. In: **Proceedings of the 10th European Conference on Computer Vision: Part II**, 2008. (ECCV '08), p. 650–663. ISBN 978-3-540-88685-3.

YUAN, C.; HU, W.; LI, X.; MAYBANK, S.; LUO, G. Human action recognition under log-euclidean riemannian metric. In: **Proceedings of the 9th Asian Conference on Computer Vision - Volume Part I**, 2010. (ACCV'09), p. 343–353. ISBN 3-642-12306-6, 978-3-642-12306-1. Disponível em: <http://dx.doi.org/10.1007/978-3-642-12307-8_32>.