UNIVERSIDADE FEDERAL DE JUIZ DE FORA

INSTITUTO DE CIÊNCIAS EXATAS

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Fábio Luiz Marinho de Oliveira

Video Motion Description Based on Histograms of Sparse Trajectories

Juiz de Fora

2016

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

Fábio Luiz Marinho de Oliveira

Video Motion Description Based on Histograms of Sparse Trajectories

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

Orientador: Marcelo Bernardes Vieira

Juiz de Fora

2016

Fábio Luiz Marinho de Oliveira

Video Motion Description Based on Histograms of Sparse Trajectories

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 5 de Setembro de 2016.

To my family and friends, the
true masters that guided me and
honored me with their love.

ACKNOWLEDGMENTS

I would like to thank my family for the unrelenting help and support, despite not having any idea of what this work is about. My mother and grandmother, Tereza and Cleonésia, are of unsurmountable value in my life. Their love and dedication to their children are something that I can only admire. My aunt Vania and my uncle Marcos, whose intellect, in their own very different ways, are sources of inspiration. My sister, Heloisa, who is tenacious, hard working, and fierce beyond what I could ever aim to be.

I'd also like to thank my friends for the companionship, laughter, and fun that I can always count on them for. Marcelo Sirimarco, for almost 25 years of friendship, no matter how close or far apart we are. Gustavo Schettino, for the example of determination and dedication. Gustavo Dias, for the example of great wit and charisma. Gabriel Duque, for the example of great sensibility, camaraderie, and dedication. Breno, for the example of great intelligence and friendship. Patrícia Duque, for the example of love and for always having her home open to us all as if it was our own.

I also would like to thank my lab colleagues, whose work I could watch firsthand and that have taught me a great deal. Ana Mara Figueiredo, Luciano Cejnog, Fernando Yamada, Liliane Almeida, João Vitor Hauck, Virgínia Mota, and specially Helena Maia, who as a true work partner, was always helpful and concerned with the success of her peers.

Finally, I'd like to thank my masters for providing me with the required knowledge and guidance. They were many, and every single one have contributed to the construction of what I am. Having done so, they are all entitled to a part of this work.

"When nothing seems to help, I would go and look at a stonecutter hammering away at his rock perhaps a hundred times without as much as a crack showing in it. Yet at the hundred and first blow it would split in two. And I knew it was not that blow that did it, but all that had gone before." Jacob A. Riis

RESUMO

Descrição de movimento tem sido um tema desafiador e popular há muitos anos em visão computacional e processamento de sinais, mas também intimamente relacionado a aprendizado de máquina e reconhecimento de padrões. Frequentemente, para realizar essa tarefa, informação de movimento é extraída e codificada em um descritor. Este trabalho apresenta um método simples e de rápida computação para extrair essa informação e codificá-la em descritores baseados em histogramas de deslocamentos relativos. Nossos descritores são compactos, globais, que agregam informação de quadros inteiros, e o que chamamos de auto-descritor, que não depende de informações de sequências senão aquela que pretendemos descrever. Para validar estes descritores e compará-los com outros trabalhos, os utilizamos no contexto de Reconhecimento de Ações Humanas, no qual cenas são classificadas de acordo com as ações nelas exibidas. Nessa validação, obtemos resultados comparáveis aos do estado-da-arte para a base de dados KTH. Também avaliamos nosso método utilizando as bases UCF11 e Hollywood2, com menores taxas de reconhecimento, considerando suas maiores complexidades. Nossa abordagem é promissora, pelas razoáveis taxas de reconhecimento obtidas com um método muito menos complexo que os do estado-da-arte, em termos de velocidade de computação e compacidade dos descritores obtidos. Adicionalmente, experimentamos com o uso de Aprendizado de Métrica para a classificação de nossos descritores, com o intuito de melhorar a separabilidade e a compacidade dos descritores. Os resultados com Aprendizado de Métrica apresentam taxas de reconhecimento inferiores, mas grande melhoria na compacidade dos descritores.

Palavras-chave: Trajetórias esparsas. Descrição de movimento. Reconhecimento de ações humanas em vídeos. Aprendizado de métrica. Histograma. Descritor tensorial.

ABSTRACT

Motion description has been a challenging and popular theme over many years within computer vision and signal processing, but also very closely related to machine learning and pattern recognition. Very frequently, to address this task, one extracts motion information from image sequences and encodes this information into a descriptor. This work presents a simple and fast computing method to extract this information and encode it into descriptors based on histograms of relative displacements. Our descriptors are compact, global, meaning it aggregates information from whole frames, and what we call self-descriptors, meaning they do not depend on information from sequences other than the one we want to describe. To validate these descriptors and compare them to other works, we use them in the context of Human Action Recognition, where scenes are classified according to the action portrayed. In this validation, we achieve results that are comparable to those in the state-of-the-art for the KTH dataset. We also evaluate our method on the UCF11 and Hollywood2 datasets, with lower recognition rates, considering their higher complexity. Our approach is a promising one, due to the fairly good recognition rates we obtain with a much less complex method than those of the state-of-the-art, in terms of speed of computation and final descriptor compactness. Additionally, we experiment with the use of Metric Learning in the classification of our descriptors, aiming to improve the separability and compactness of the descriptors. Our results for Metric Learning show inferior recognition rates, but great improvement for the compactness of the descriptors.

Keywords: Sparse trajectories. Motion description. Video human action recognition. Metric learning. Histogram. Tensor descriptor.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

S, s	Scalar value, $S \in \mathbb{R}$ used for fixed values or parameters, $s \in \mathbb{N}$ used as indices.
$\mathbf{r}$	Discrete coordinate or vector $\mathbf{r} = (x, y)$, with $x, y \in \mathbb{N}$.
$\mathcal{D}$	Set of any kind. Example: $\mathcal{D} = \{d_1, d_2, ..., d_n\}$.
$\mathbf{T}$	Tensor or other Positive Semi-definite matrices.

LIST OF ACRONYMS

4SS Four Step Search

BM Block Matching

BoF Bag-of-Features

BoW Bag-of-Words

DCS Divergence-Curl-Shear

HAR Human Action/Activity Recognition

HOF Histogram of Optical Flow

HOG Histogram of Oriented Gradients

HOG3D Histogram of Oriented 3D Gradients

MBH Motion Boundaries Histogram

OSR Object Scene Recognition (dataset)

PSD Positive Semi-definite (matrix)

SAD Sum of Absolute Differences

SSD Sum of Square Differences

SVM Support Vector Machine

VSBM Variable Size Block Matching

CONTENTS

1 INTRODUCTION

Movement description has been a challenging and popular research field over many years. It is an area within computer vision and signal processing, but is also very close to machine learning and pattern recognition. The challenges it poses in all of these disciplines, along with its many potential applications draw the attention of researchers. It can be applied to many real-life, human-centric problems like surveillance, health-care, and sports performance analysis, as well as to video retrieval in databases, considering the growth and popularity of recording devices. It is also a field that encompasses, and very frequently is closely related, to Human Action Recognition (HAR), which is an area that provides a plethora of databases and successful research to validate and compare one's work. Very often, movement description is used to recognize simple human activities, like running, jumping, driving, etc. Complex activities require a certain level of abstraction or semantic interpretation, like assaulting, playing, greeting, etc. These complex activities are especially hard to recognize using only motion information from a video, even though motion is an important lead, and thus, are beyond what we aim to achieve in this work.

A very widespread procedure for activity recognition is to generate a collection of descriptors (or feature vectors) from the videos, and use these descriptors as input for a machine learning technique which classifies or separates them according to the action portrayed, be it a Support Vector Machine (SVM) (CORTES; VAPNIK, 1995), Artificial Neural Networks (ANN), or any other classifier. In this work, we are interested mainly on the first part of this framework, that is, how to generate a good description for the movement in the video. We understand that this focus on describing movement in videos is a justified effort in and of itself. Over the years, many different descriptors were proposed, like Histogram of Oriented Gradients (HOG) (DALAL; TRIGGS, 2005), Histogram of Optical Flow (HOF) (DALAL et al., 2006), Motion Boundary Histograms (MBH), Divergence-Curl-Shear (DCS) (JAIN et al., 2013) with a multitude of variants and combinations among each other.

These descriptors all rely on a characteristic of digital video representation. They all compare adjacent frames of a video to obtain a form of gradient or flow of the brightness values between them. Whether it is a very fine, pixel to subpixel-wise flow, like gradients or optical flows, or a coarser one, subimage-wise, like our choice for this work: block matching. The intuition behind these brightness comparisons is that the amount of time that passes between two adjacent frames is little enough that it is safe to assume that similar brightness configurations represent the same object or region in the scene. This intuition is massively supported by the aforementioned works, considering they present great results for their applications, to the point that it becomes a fundamental premise in our work.

We also draw other premises from what has been evidenced by the works of Caetano (2015) and Figueiredo (2015). Firstly, trajectories are very valuable to describe motion, notably the use of dense trajectories by Wang et al. (2011). They provide richer movement information than merely aggregating the result of comparisons of pairs of frames. They allow for a greater coverage of the scene and its moving elements. A second assumption we derive mainly from Mota et al. (2013) and Zelnik-Manor and Irani (2001) is that tensors are able to synthesize the gathered movement information into a concise form, keeping similar elements close in tensor space. We can also expect that, as it has been heavily evidenced in both the state-of-the-art (WANG et al., 2013a) and our own recent works (OLIVEIRA; VIEIRA, 2015; FIGUEIREDO et al., 2016), in conjunction with histograms, block matching provides a fast computing, compact motion descriptor.

1.1 PROBLEM DEFINITION

The problem in this dissertation is to generate a descriptor to represent motion in image sequences based in trajectories of elements that are invariant with respect to brightness. This is a very harsh and brief definition of the problem, which we will clarify and detail below.

The above mentioned *invariance with respect to brightness* is rooted in optical flow (GIBSON, 1950), and how the human vision perceives relative motion between objects in view and the viewer itself. Horn and Schunck (1981), one of the earlier works with regards to optical flow within the computer vision context, highlights how the apparent movement of brightness patterns are important to determine the spatial arrangement and changes in such arrangement for the objects in the scene.

The importance of the *trajectories* of said elements is also noted by Jain and Jain (1981), when concerning video coding and compression, state that *"In practice, a signifi-*

cant component of the motion in a scene can be approximated by piecewise translation of several areas of a frame with respect to a reference frame." (JAIN; JAIN, 1981, p. 1799)

In the context of this work, just as in Jain and Jain (1981), these elements are blocks, namely subimages, in which the frames are partitioned. These blocks supposedly keep the same disposition of pixel brightness values throughout a sequence of frames, hence the name *block matching*. It is not required that these blocks are found in the same position on subsequent frames, though. Quite on the contrary, in order to track motion it is expected the blocks are found on distinct positions as we advance on the sequence of frames, constructing the trajectories.

The solidity of these works, in their respective fields, gives rise to one hypothesis of this dissertation: that *trajectories consisting solely of block matching information can produce a motion descriptor with efficiency and efficacy.* By efficiency we mean the trait of a descriptor that is both compact and fast computing. And by efficacy we mean the characteristic of a descriptor of being representative of the motion and suitable for use in application scenarios.

By requiring these properties of the descriptor, we are confronted with another problem, concerning the *motion representation* part of the definition. We must synthesize the information from the trajectories into a compact form, given they alone represent too much data to achieve the desired compactness. But, at the same time, a complex processing to reduce the size of the descriptor, like a dictionary or bag-of-words approach employed in (WANG et al., 2013a), would work against our other desired property of fast computation. From previous research experience (OLIVEIRA; VIEIRA, 2015), aggregating the trajectories directly into histograms has been a viable choice. Therefore we arrive at another hypothesis: that is, *synthesis/dimension reduction scheme based on histograms of relative displacements is able to aptly describe motion in a frame sequence.*

In order to address the validation of the descriptor, by assessing how it fares in Human Action Recognition, we venture into another problem, and a machine learning one, more specifically that of classification. That is, given a population of samples divided into classes (or categories), the system must identify to which class a new sample belongs. This is done using the set of samples with known class assignments, named training set. In this work, the samples for the classification problem are the video descriptors and the classes describe actions portrayed in these videos, and the classifier of choice is a Support

Vector Machine (CORTES; VAPNIK, 1995).

Still in this matter of classifying the descriptors, we also employ a Metric Learning method, as we hypothesize that it improves the separability of the data to be classified. We are unable, in this work, to confirm or deny this hypothesis, although we still provide data on the related experiments in Chapter 6.

To summarize, the three hypotheses of this work are:

- Trajectories consisting solely of block matching information produce a motion descriptor with efficiency and efficacy;
- Histograms of relative displacements are able to aptly and synthetically describe motion in a frame sequence;
- Metric Learning improves the compactness and separability of motion descriptors in feature space for classification.

1.2 OBJECTIVES

The main objective of this work is to describe apparent motion adequately and efficiently, that is, to obtain a compact, low computational cost, effective descriptor for motion. Remark that this objective is very much in line with one of the hypotheses of this work. By pursuing a descriptor designed with these characteristics, we are able to produce evidence, through experimental results, to substantiate our hypothesis.

As a secondary objective and a way to validate our descriptor, we evaluate its use in the context of Human Action Recognition. We obtain motion descriptors that are both competitive, in terms of recognition rates for HAR, and fast computing.

To achieve this objective, we make several choices for simplicity's sake. As we'll explore more in the related works chapter, many authors, including those whose works are the state-of-the-art, employ pyramidal sampling, multiple feature descriptors, fine granularity flows, object of interest segmentation, local features, to cite a few. All of these approaches, although being able to achieve very high recognition rates, are very costly concerning computational resources.

As another secondary objective, we also aim to assess the influence of Metric Learning, specifically the quadruplet-wise method proposed by Law et al. (2013), in the context of Video Human Action Recognition. For that matter, we explored the inclusion of it right before the classification part of our process, as a way of improving the compactness and possibly the separability of the descriptors.

2 RELATED WORKS

The descriptors by Wang et al. (2011) are based on trajectories of feature points. These feature points are densely sampled and tracked based on displacement information from the dense optical flow algorithm of Farnebäck (2003) on multiple spatial scales. Local descriptors are then computed within space-time volumes around trajectories using HOG, HOF, MBH, or a combination of them. These local descriptors are used to build a codebook in a standard bag-of-features approach. Their work presents very high recognition rates for the three datasets used in our evaluation, KTH, UCF11 and Hollywood2. Wang et al. (2013b) later improve on this method, using camera motion estimation, a human detector, and a different encoding scheme. Wang et al. (2013b) also use three other datasets: HMDB51, Olympic Sports, and UCF50. The descriptors built with our method differ from these ones cited on several of aspects. Our descriptor is global, that is, the information that composes it comes from whole frames, and not just close vicinities of a point of interest or feature. Moreover, our descriptor for a given sequence does not rely on information from the descriptors of other sequences, since no bag-of-words, Fisher Vectors or any other similar encoding is employed during the descriptor construction[1]. This independence from information from other sequences makes our descriptor what we call a self-descriptor. All the necessary information to describe the motion in a scene is assumed to be contained solely in the scene itself.

Wang et al. (2015) propose a novel video descriptor, called trajectory-pooled deep-convolutional descriptor (TDD), which the authors regard as benefiting from both hand-crafted features and deep-learned features. The hand-crafted features extracted are dense trajectories from Wang et al. (2013b) combined with Fisher Vectors (SANCHEZ et al., 2013). The deep-learned features are obtained through Convolutional Networks (ConvNets). Their results show that TDDs outperform previous state-of-the-art hand-crafted features and deep-learned features in the datasets used: HMDB51 and UCF101.

Tensors are powerful tools to agglomerate information, that have been used in many different works in the pattern recognition area over recent years, especially within the area

[1]The Metric Learning procedure used in part of this work contradicts that, but the descriptors themselves are independent from each other. Moreover, being still an incipient aspect of our research, the Metric Learning approach is complementary in this work, and not its main contribution.

of this work, that focus on describing motion. Mota et al. (2012) use optical flow projected on Legendre polynomials and aggregate the resulting coefficients into orientation tensors. This approach produces a global descriptor. In this work, as mentioned, the descriptor is also global and the descriptor is also represented as a tensor. However, a scheme more similar to ours is presented in Perez et al. (2012). In their work, three dimensional HOG (HOG3D) (KLÄSER et al., 2008) is used to extract motion information between frames, then the vectors are aggregated into two dimensional histograms, and finally the histograms are combined into orientation tensors. The work of Perez et al. (2012) differs from ours regarding its use of very fine flow of brightness (HOG3D), three dimensional tensors, and the lack of trajectories. Mota et al. (2013) also employ HOG and aggregate its results using both histograms and tensors, but many tensors are computed for each frame, which are partitioned in a grid structure. The final descriptor for a frame is the concatenation of the tensor descriptors computed in each cell of the grid.

Tensors have also been recently used in action recognition using 3D skeleton representations, instead of RGB camera information, as in the work by Koniusz et al. (2016). The authors use tensors to capture higher-order relationships between skeleton joints, modeling both the compatibility between joints in one sequence with joints in another sequence, and the action dynamics of a sequence. Their tensors computed from these models are classified using an SVM. Such a work has little practical connection with our own, but it serves to indicate how tensors can be valuable to encode information in the motion description field.

Two other works make use of both trajectories and tensors, the ones by Caetano (2015) and Figueiredo (2015). Caetano (2015) extracts dense trajectories much like Wang et al. (2011), but proposes a descriptor based on the cross product between the local trajectory displacement and gradients on surrounding windows. Additionally, the trajectories by Caetano (2015) are clustered by shape using the *k-means* algorithm. That is, they are represented as vectors of angles between displacement vectors composing the trajectory. The author argues that this makes the trajectories comparable regardless of their direction in the frame. The author also argues that by doing so, camera movement is easily detected and instead of being discarded, it is instead represented in one of the clusters, adding to the information about the scene. The final descriptor in his method is the concatenation of the tensors for each cluster. The work by Figueiredo (2015) also follows this scheme from Caetano (2015), with the cross product, shape representation, trajectory clustering, and final tensor representation. But instead of dense trajectories, the author extracts sparse trajectories using block matching, exploring three different approaches to this extraction.

Jain et al. (2013) propose a decomposition of visual motion information into *dominant* and *residual* to improve action recognition algorithms. Their method works on top of existing techniques in the area, so dense trajectories, HOF, HOG, and MBH are used to obtain the motion information. They also propose a new motion descriptor, *Divergence Curl Shear* (DCS), and explore the combination of their novel descriptor with those already available.

Another method that is compatible and complementary to local motion and appearance based methods is the one by Fernando et al. (2015). But instead of relying on the apparent movement of elements in a scene, it uses a rank machine to sort the frames of a scene temporally by their appearance. The parameters of the ranking functions learned serve as the new video representations for classification. The authors argue that a ranking function capable of sorting the frames is also able to capture the evolution of appearance in the video. They also consider their own method to be easy to interpret and implement, fast to compute, and effective in recognizing different kinds of actions. In their evaluation, they use diverse datasets, *Hollywood2* and HMDB51 for generic actions, MPII-cooking activities for fine-grained actions and *Chalearn* for gestures.

The work of Jain et al. (2015) also takes an alternative path to action recognition. Instead of using motion as the key information, they detect known objects and use them as indicative of which action is being performed. The authors show, through their empirical study, that objects are semantically relevant for actions. Their combination of objects and motion information also improves the state-of-the-art for both action classification and localization. To detect objects, they compute the likelihood of the object categories being present in each frame of the videos using a convolution network. The object categories are built with samples from the ImageNet dataset, amounting to 15000 object categories.

Other works in the area include those by Schuldt et al. (2004), Liu et al. (2009) and Marszalek et al. (2009), which not only propose their own action recognition methods, but also made available the KTH, UCF11, and *Hollywood2* datasets, respectively.

Related works concerning Metric Learning are mentioned along with its fundamentals in Section 3.3, to present its context and its variety of applications.

3 FUNDAMENTALS

This chapter provides some basic concepts with the purpose of allowing a better understanding and clarifying some terminology of the following chapters of this work.

3.1 BLOCK MATCHING

Block Matching (JAIN; JAIN, 1981) is a method for tracking elements along sequences of images. It was conceived in the context of video encoding and compression, with the objective of minimizing the amount of data sent through the network of a video conference system or stored in a video file. This is done by partitioning a frame into blocks, then estimating the displacement of each individual block by the comparison of its brightness values with those in a neighborhood of the block in the following frame. Having this estimate, instead of transmitting (or storing) whole frames, one frame could be transmitted whole, as a reference, and then only relative displacements along with very distinct portions of following frames would be necessary to represent them. In order for this scheme to work properly, there is the assumption that if there is continuous motion in an image sequence, several blocks from one frame can be found in the next one, but in slightly different positions.

In more formal terms, the process consists in partitioning a frame $f^{(n)}$ into M rectangular blocks and finding, for each block, the displacement vector tuple

$$(\mathbf{d}^{(1)}, \mathbf{d}^{(2)}, ..., \mathbf{d}^{(S)})_m^{(n)}, \ m \in [1, M], \tag{3.1}$$

corresponding to its trajectory in a sequence of consecutive frames $(f^{(n)}, f^{(n+1)}, ..., f^{(n+S)})$ of a video, where S is the size of the trajectory. This size could vary in other applications, from block to block being tracked. However, in this work, S has a fixed predetermined value. This tuple is obtained by finding the neighboring block that minimizes some error function ϵ, so

$$\mathbf{d}^{(t)} = (b_x^{(t+1)} - b_x^{(t)}, b_y^{(t+1)} - b_y^{(t)}) \ s.t. \ \min_{\mathbf{b}^{(t+1)}} \{\epsilon(\mathbf{b}^{(t)}, \mathbf{b}^{(t+1)})\}, \ t = 1, ..., S. \tag{3.2}$$

In the equation above , $\mathbf{b} = (b_x, b_y)$ denotes the coordinates of a block b in the frame and the error function ϵ takes the coordinates of two blocks to compare and returns a single real value. Should the blocks at $\mathbf{b}^{(t)}$ and $\mathbf{b}^{(t+1)}$ have similar brightness configurations, the value of ϵ will be near zero and the displacements will provide a good representation of the motion along the frame sequence.

There are two block matching components that call for discussion: *error function* and *search strategy.* The error function is the sole criterion for block matching to judge the similarity between two blocks. In this work, we employ the Sum of Absolute Differences (SAD) as the error function. Although other error functions could have been used, like Mean Absolute Differences (MAD) or Normalized Cross-Correlation (NCC), we chose SAD, as it is the simplest and fastest. Moreover, our previous research experimentations indicate that the choice of error function causes little to no impact on the performance of our descriptors. The SAD is defined as:

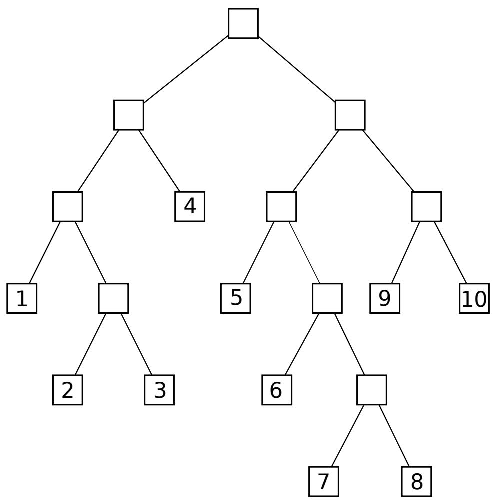
$$D_{SAD}(\mathcal{B}) = \sum_{x,y \in \mathcal{B}} |f^{(n)}(x,y) - f^{(n+1)}(x,y)|, \tag{3.3}$$

where $f^{(n)}(x,y)$ is the brightness value of coordinates (x,y) in frame f and $\mathcal{B}$ is the set of pixel coordinates in a given block.

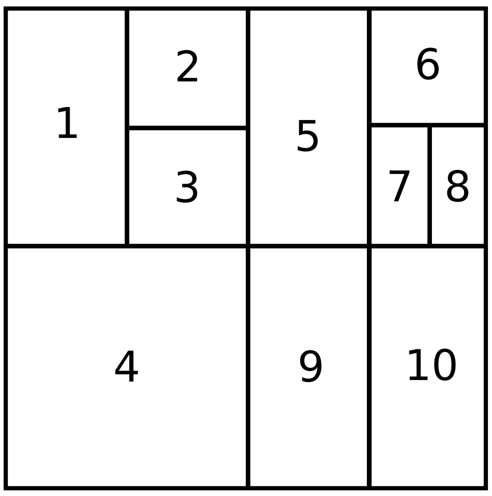
The search strategy is the part of block matching most responsible for its fast computation. It is the sequence of steps the block matching procedure takes to find the best match for a block, that is, the block in the next frame with the least error function value. Searching a whole frame for a block is prohibitively expensive in terms of computation time. So the search is restricted to a window, a neighborhood around the block to be matched. This is done under the assumption that objects in the scene move continuously. Even so, the search window alone is not enough to optimize the speed of block matching. So a sequence of steps, usually a gradient descent-like procedure is also used. A number of approaches for this procedure have been proposed over the years, like Li et al. (1994), Po and Ma (1996), and Zhu and Ma (2000). In this work, we chose the Four Step Search (4SS) (PO; MA, 1996) due mainly to its fast computation and easy implementation. 4SS is a gradient descent based search strategy which limits the amount of computation to find optima in a close vicinity. Even though the use of a search strategy has a great impact on the running time of block matching, the choice of which search strategy to use seems to have no influence in the descriptor performance, as we could also investigate in our past experience.

3.1.1 VARIABLE SIZE BLOCK MATCHING

Recognizing the advantages of block matching for compression, Puri et al. (1987) and later Chan et al. (1990) developed it even further by proposing *variable size block matching* (VSBM). Both the conventional block matching and VSBM start out the same way, with the partition of the frame into equally sized blocks. But instead of keeping this configuration throughout the whole computation, VSBM blocks can be split into smaller blocks or merged into bigger regions, possibly even non-rectangular ones. The criteria for when to split or merge blocks are varied, but are often related to the matching error, the displacement vector, or some measure over brightness intensities, like in picture segmentation (HOROWITZ; PAVLIDIS, 1976), where the variance of the brightness is taken into account. The blocks can be split in two or four new blocks, giving rise to a bin- or quad-tree structure, where leaf nodes are blocks of varied sizes (Figure 3.1). Ideally, the block boundaries would coincide with those of moving objects in the scene, and thus provide a good approximation of the optical flow, but at a coarser granularity.



(a) Bin-tree structure



(b) Corresponding frame partition

Figure 3.1: Block matching bin-tree structure and corresponding partition of a frame. Figure adapted from Chan et al. (1990).

In our implementation, for the sake of simplicity and lower computational cost, only split operations are allowed and they are carried out when the matching error is above a fixed threshold parameter. That is, after finding the best match for a block, if the matching error is above said threshold, the block is split in smaller blocks and the matching is computed for them. If the matching error for any of these smaller blocks is still above the error threshold, the process continues recursively, by splitting the block into two and attempting to match the new ones. This recursion stops when the error is below the threshold or the sizes of the blocks reach a pre-established minimum size. Even though we do not implement merge operations, if the sum of the matching errors for two new blocks is larger than the matching error of the original block, we allow for an ”undo”, and keep the original block instead of the split smaller blocks.

3.2 ORIENTATION TENSOR

Orientation tensors are a particular type of tensor, that have properties that allow for the representation of a three-dimensional surface and its orientation in space. In the work of Knutsson (1989), it is shown how to map local brightness correlations into an orientation tensor, that is, 1-D lines (or similar structures) into a 3-D surface. As Knutsson (1989) puts, a suitable mapping should meet three basic requirements: *uniqueness*, *uniformity*, and *polar separability*. This is specially useful in this work when it comes to building the final descriptor. More on that can be found in Section 4.4. We'll now focus on the tensor itself, summarizing its definition, terminology, and some properties, as discussed in depth in the works by Knutsson (1989) and Westin (1994).

Let $\mathcal{V}$ be a linear space with its dual denoted as $\mathcal{V}^*$. A tensor of order (p, q) may be defined as a multi-linear mapping, that is, a mapping that is linear in each of its arguments:

$$\mathbf{T} : \underbrace{\mathcal{V}^* \times \mathcal{V}^* ... \times \mathcal{V}^*}_{p} \underbrace{\times \mathcal{V} \times \mathcal{V} ... \times \mathcal{V}}_{q} \rightarrow \tag{3.4}$$

From this definition, a vector is a $(1, 0)$ tensor, and $(2, 0)$, $(0, 2)$, and $(1, 1)$ tensors can be represented by matrices. Note that since p and q determine the number of indices, the order of a tensor is not related to the dimension of the vector. A tensor of order $(2, 0)$ can be represented by a 2×2, a 3×3 or even larger matrix, for instance.

As for the mapping and its required properties we have:

- Uniqueness:

$$\mathbf{T}(\mathbf{x}) = \mathbf{T}(-\mathbf{x}), \tag{3.5}$$

  where $\mathbf{x}$ is a vector in the original space and $\mathbf{T}$ is the map of $\mathbf{x}$.

- Uniformity:
$$\|\delta\mathbf{T}\| = c\,\|\mathbf{x}\|_{r=const.}\,, \tag{3.6}$$
where δ denotes first order approximations $r = \|\mathbf{x}\|$ and c is a "stretch" constant, meaning that the mapping locally preserves the angle metric of the original space.

- Polar separability:
$$\|\mathbf{T}\| = f(\|\mathbf{x}\|), \tag{3.7}$$
where $f : \mathbb{R} \rightarrow \mathbb{R}$, meaning the norm of $\mathbf{T}$ is independent of the direction of $\mathbf{x}$.

A mapping that maps the vector $\mathbf{x}$ onto the tensor $\mathbf{T}$, all the while presenting the above properties is given by:
$$\mathbf{T} = \mathbf{x}\mathbf{x}^\top, \tag{3.8}$$
with its norm taken to be the Frobenius norm:
$$\|\mathbf{T}\|^2 = \sum_{ij} t_{ij}^2, \tag{3.9}$$
where t_{ij} are the components of $\mathbf{T}$.

Take special note of Equations 3.8 and 3.9, as they play an important role in our descriptor generation (Section 4.4).

3.3 METRIC LEARNING

Consider a scenario in which we are required to compute the similarity or distance between pairs of video sequences regarding their content. For that, it is important to know how to measure the distances. For instance, if the goal is to identify and match human actions, then we should choose a distance function that highlights certain features, like limb movement, interaction with other people and/or objects, etc. If the goal were to identify videos portraying natural landscapes, we would clearly rely on other features, like presence of trees or sky visibility. Instead of determining by hand an appropriate distance function for each of these cases, we could employ metric learning, which is an automated and supervised way to learn these task-specific distance metrics. The goal of metric learning is to find a matrix for a Malahanobis-like distance that best emphasizes whatever features are important.

Plenty of works have contributed over recent years, with a vast array of variants, extensions, and applications for metric learning. Possibly the most well known work is that of Xing et al. (2003), which poses the problem as a convex optimization problem. Their main application was to improve clustering by learning the distance function prior to clustering the data. Schultz and Joachims (2004) take an approach similar to the soft-margin SVM problem, in which by choosing the metric matrix to be diagonal, it becomes a vector of feature weights. The optimization then involves a quadratic objective over the feature weights, along with the hinge loss applied to a linear function of the feature weights. Weinberger and Saul (2009) proposed one of the most popular metric learning methods: Large Margin Nearest Neighbors (LMNN). The intuition behind it is that a data point should have the same label as its nearest neighbors, which is enforced by a relative distance with margins between points with different labels. Davis et al. (2007) method, named Information-Theoretic Metric Learning (ITML) introduce a regularizer which provides a distance function with scale and translation invariance, along with some other properties. Law et al. (2013) formulate metric learning based on quadruplets instead of pairs, to better assess the relations of similarity and dissimilarity. Their applications include image retrieval and an analysis of web page contents disposition. Other works containing metric learning in computer vision include: Guillaumin et al. (2009) for face recognition, and Tran and Sorokin (2008) for action recognition. Other general applications are: text analysis (DAVIS; DHILLON, 2008), music analysis (SLANEY et al., 2008) and even program debugging (HA et al., 2007).

The metric learning formulation made popular by Xing et al. (2003), consists in obtaining a Malahanobis-like distance:

$$D^2_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{W}(\mathbf{x}_i - \mathbf{x}_j), \mathbf{W} \succeq 0, \tag{3.10}$$

where $\mathbf{W} \in \mathbb{R}^{d \times d}$ is a symmetric positive semi-definite (PSD) matrix ($\mathbf{W} \succeq 0$) and $(\mathbf{x}_i - \mathbf{x}_j) \in \mathbb{R}^d \times \mathbb{R}^d$ are representations of the objects one wants to measure distances between.

In pairwise approaches, like Xing et al. (2003), $\mathbf{W}$ is learned by optimizing $D^2_{\mathbf{W}}$ on a training set partitioned into subset $\mathcal{S}$, of similar points, and subset $\mathcal{D}$, of dissimilar points.

The objective function is then defined as:

$$\min_{\mathbf{W}} \sum_{i,j \in \mathcal{S}} D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) \quad s.t. \sum_{i,j \in \mathcal{D}} D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1, \mathbf{W} \succeq 0, \tag{3.11}$$

as a way to minimize the distance between similar points, while keeping dissimilar points away by a margin. This is done to support clusterings, classifications, rankings, and any other problem whose solution benefits from differentiating samples or instances through a dissimilarity function.

3.3.1 Q-WISE METRIC LEARNING

A metric learning extension and its application in computer vision deserve special attention. The quadruplet-wise formulation for metric learning (LAW et al., 2013), based on the relative attributes approach (PARIKH; GRAUMAN, 2011), has not yet been thoroughly explored in activity recognition (ZHANG et al., 2015). As claimed by Law et al. (2013), it is more useful to assess the similarity as *two images are more similar than two other images* relations, rather than binary labels establishing that they are considered equivalent. For that matter, quadruplet-wise approaches compare pairs of similarities (D_{ij}, D_{kl})[1], which in turn require four different points $(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l)$.

The Malahanobis-like metric matrix $\mathbf{W}$, in Q-wise Metric Learning can be learned in one of two contexts, which Law et al. (2013) bring to attention, to avoid both an expensive computation and overfitting:

- $\mathbf{W}$ is diagonal: $diag(\mathbf{W}) = \mathbf{w}$, so that $\mathbf{W}$ is PSD iff $\mathbf{w} \geq \mathbf{0}$. In this case the metric acts as a scaling of the components of $\mathbf{x}$.
- Optimization over rows: $\mathbf{W} = \mathbf{L}^\top \mathbf{L}$, with $\mathbf{L} \in \mathbb{R}^{c \times d}$, with possibly $c \leq d$. In this case, each one of the c rows of $\mathbf{L}$ represents an attribute, and can be learned independently.[2]

In both cases, we can refer to the parameters to be learned as a vector $\mathbf{w}$. Let $\mathbf{z}_q$ be the vector of differences of the quadruplet $q = (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l)$, or $\mathbf{z}_q = \mathbf{z}_{ijkl}$. So $D_{\mathbf{W}}(\mathbf{x}_k, \mathbf{x}_l) - D_{\mathbf{W}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{w}^\top \mathbf{z}_q$. These quadruplets compose two sets of constraints, $\mathcal{A}$

[1]The notation D_{ij} in this section is merely shorthand for $D(\mathbf{x}_i, \mathbf{x}_j)$

[2]This second case is the one that draws our interest, as we adapt this method to make a classification. This is further discussed in Section 5.5.

and $\mathcal{B}$, with the goal of learning the parameters of $D_{\mathbf{w}}$ satisfying the maximum number of said constraints.

$$\forall(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l) \in \mathcal{A} \mid D_{\mathbf{w}}(\mathbf{x}_k, \mathbf{x}_l) \geq D_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_j) + 1 \tag{3.12}$$

$$\forall(\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l) \in \mathcal{B} \mid D_{\mathbf{w}}(\mathbf{x}_k, \mathbf{x}_l) \geq D_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_j) \tag{3.13}$$

The constraints on set $\mathcal{A}$ consider a margin between samples, while the ones in $\mathcal{B}$ do not. The objective function is then defined as:

$$\min_{\mathbf{w}} \sum_{q \in \mathcal{A}} L_1^h(\mathbf{w}^\top \mathbf{z}_q) + \sum_{q \in \mathcal{B}} L_0^h(\mathbf{w}^\top \mathbf{z}_q) + \lambda \|\mathbf{w}\|_2^2 \tag{3.14}$$

where L_1^h is a differentiable approximations of the hinge loss when the parameter $h \to 0$, inspired by the Huber Loss function. L_0^h is an adaptation of L_1^h without the safety margin. The last term is a regularization term tuned by a parameter λ to avoid overfitting.

On the matter of obtaining constraints, the work of (LAW et al., 2013) is largely based on the relative attributes from (PARIKH; GRAUMAN, 2011). This approach consists on loosely ordering quadruplets according to the presence of a desired attribute. Let $q = (\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \mathbf{x}_l)$ be a quadruplet with samples from four different classes: $\mathbf{x}_i \in \mathcal{E},\ \mathbf{x}_j \in \mathcal{F},\ \mathbf{x}_k \in \mathcal{G},\ \mathbf{x}_l \in \mathcal{H}$. Also let the presence of an arbitrary attribute be such that the classes are ordered as follows: $\mathcal{E} \prec \mathcal{F} \prec \mathcal{G} \prec \mathcal{H}$. Then it follows that $q \in \mathcal{A}$ as a constraint $D_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_l) \geq D_{\mathbf{w}}(\mathbf{x}_j, \mathbf{x}_k) + 1$. Were the classes $\mathcal{F}$ and $\mathcal{G}$ somewhat indistinguishable in terms of the attribute in question, the ordering would be $\mathcal{E} \prec \mathcal{F} \sim \mathcal{G} \prec \mathcal{H}$ and the constraint $q \in \mathcal{B}$, such that $D_{\mathbf{w}}(\mathbf{x}_i, \mathbf{x}_l) \geq D_{\mathbf{w}}(\mathbf{x}_j, \mathbf{x}_k)$. Notice that the first and last samples $(\mathbf{x}_i, \mathbf{x}_l)$ are less similar than the second and third ones $(\mathbf{x}_j, \mathbf{x}_k)$ according with the ordering, and thus the former pair have a greater distance between themselves than the latter pair.

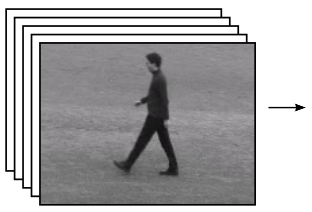
Once the matrix $\mathbf{W}$ is learned, more specifically $\mathbf{L}$, the samples $\mathbf{x}_i$ can be represented through the presence of each relevant attribute, in a high level representation: $\mathbf{h}_i = \mathbf{L}\mathbf{x}_i$. This representation is interesting in our work, since it greatly improves the compactness of our descriptors. More in Section 6.2.

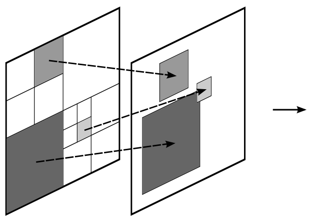
This approach could be translated to the video context, by comparing actions instead of image characteristics. Even though it is hard to determine or quantify whether a certain action is being performed more or less, the intuition behind the quadruplet relations still holds. That is, it is easier to assess how similar is a pair in relation to another pair of videos. The introduction of the paper from Law et al. (2013) provides a brief and clear explanation using static images and the presence of ”smile” in them. An adaptation of the same method to HAR is proposed in this work in Section 5.5.

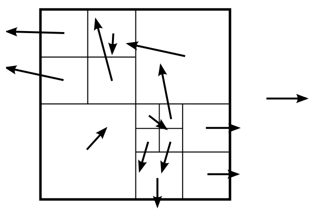
4 PROPOSED METHOD

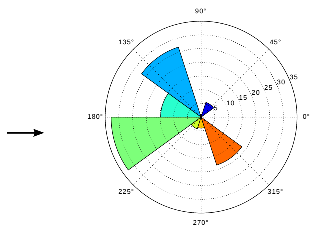
In this chapter, we present the method implemented throughout the research and its specifics. Section 4.1 provides an overview of the proposed method. Section 4.2 explores the relationship between elements of our method and the hypotheses of this work. Some of the concepts and how we address them have been already noted previously in this dissertation, especially in Chapter 3. The following sections: Section 4.3, Section 4.4, and Section 4.5 focus on details for each element of our method and how they are tied together.

4.1 METHOD OVERVIEW

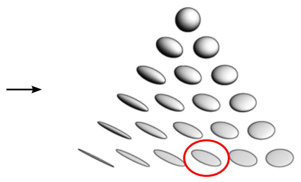

(a) Frame sequence.


(b) Block displacements.


(c) Block matching result.



(d) Histogram.


(e) Tensor representation.

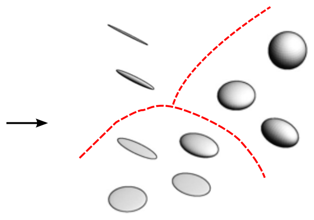

(f) Classification.

Figure 4.1: Method overview. Each part of our method is represented in a simplified manner, mainly to illustrate how straightforward the approach is. (a) shows a sequence of frames. (b) shows the computation of block matching displacement vectors. (c) illustrates a frame with its displacement vectors. Trajectories are build by matching blocks iteratively for each pair of successive frames in the sequence. (d) shows the histogram of directions for a single frame. (e) shows the tensor representation for the histogram circled in red. (f) depicts a classification scheme where each region separated by the dashed curves possibly represent a different action in the sequence.

Figure 4.1 shows an overview of the method. It highlights the simplicity of the method, meaning how straightforward the approach is. First, the collection of frames (Figure 4.1(a)) from which the trajectories are extracted. In Figure 4.1(b) and Figure 4.1(c), the block displacements between a pair frames are shown. The trajectories are built by concatenating the vectors outputted by successive block matchings. Then, the trajectories are aggregated into a histogram descriptor (Figure 4.1(d)) for each frame. The histograms then assume a tensor representation. In the figure, many are shown, but the one that corresponds to the histogram depicted is the one circled in red in Figure 4.1(e). The tensors are summed up, so there is a single tensor for the whole sequence[1]. Finally, for the sake of validating our descriptors, they undergo a classification (Figure 4.1(f)). We expect that similar actions depicted in a video are reflected as tensors nearby each other in the classification space, as the figure suggests.

4.2 METHODOLOGY

This section presents how the hypotheses were elaborated throughout the development of this work and what were the main questions that arrived.

Our first hypothesis has two main aspects we must address: block matching and trajectories.

Block matching comes from works in the video encoding and compression field. Research about it is rooted in the dissemination of media content, specially television, back in the 1970's decade. It has been a topic of interest ever since, particularly now, that the Internet has taken place as the main instrument of this dissemination of video content.

To conceive this work, however, we were interested in investigating how block matching could be used as a low-cost flow of brightness patterns. We surmise that most of the works which also have the objective of describing apparent motion, employ some form of flow that is highly demanding, in terms of computational resources. These often are based on gradients, feature point matching, or optical flow. Not only that, but pyramidal sampling of the frames, background subtraction, bag-of-words and other techniques are frequently used. These all tend to greatly decrease the speed of computation.

The rationale behind the use of trajectories was much simpler. We just aim to follow the state-of-the art of the human action recognition field, which extensively employs dense trajectories. Once again, guided by simplicity and efficiency, we confronted the question of how to build a lower cost trajectory scheme.

[1]The tensors are depicted as ellipsoids for illustrative purposes. The actual tensor dimension is much greater than 3.

An important remark is that this regulating idea of simplicity, efficiency and low cost of computation is not desultory. We account, through the available literature for motion description, that a great deal of works, employ an array of techniques and methods with little to no regard to their complexity. In fact, very few works report any kind of speed or efficiency metrics, or at least take into consideration using simpler, faster, more scalable methods. Although we are not unaware of the high quality of these works, we consider this lack of efficiency evaluation to be hurtful to their application capabilities.

With all that in mind, we decided to go in the alternative direction of approximating the quality of other motion descriptors, while having a much simpler body of procedures. Hence the first hypothesis of this work.

The hypothesis concerning histograms was conceived considering how there is a consensus that histograms are great dimensionality reduction tools. This is evidenced by several works of great relevance in the motion description area that use histograms. Notably Dalal and Triggs (2005) and Dalal et al. (2006), whose contribution is part of many of the works in the area ever since.

Histograms of direction are discriminative of the exact information of our interest: orientation, direction. They also have the added benefit of always having the same size, regardless of how many trajectories we have, or how long they are. Furthermore, using trajectories directly as the descriptor or directly into tensors would lead to a final descriptor orders of magnitude bigger than the one we currently compute using histograms. On top of that, this approach with no regard to the compactness of the final descriptor could still provide only a marginal gain, or even no gain at all, in terms of motion representation.

Histograms have also been of great avail in works within our research group, like in Perez et al. (2012) and Mota et al. (2013). So, for all these reasons, we decided to investigate their use in our own context of motion description, using block matching displacements.

The idea of using Metric Learning influences the validation part of our method. For this validation, we employ some kind of machine learning technique to differ between descriptors of each type of action portrayed in the sequences. In other words, the descriptors are classified into action categories. In order to classify these descriptors, there must be a way to measure the similarity (or dissimilarity, or distance) between each other. Our descriptors are essentially vectors in a high-dimensional Euclidean space, and we assume that vectors are nearby each other whenever they represent the same kind of motion. So we turned to Metric Learning as a way to capture the disposition of the vectors and build a distance function for said space in a way that the similarity/dissimilarity relations are highlighted.

4.3 TRAJECTORIES

The extraction of the trajectories is the first and most visually oriented part of our method. We construct the trajectories as described in Equations 3.1 and 3.2 in Section 3.1. First, a frame is partitioned into blocks of equal size. Then, between the first two frames of the trajectory, a new configuration of blocks emerge through VSBM. From the second frame onwards, until the last frame of the trajectory, the sizes of the blocks remain the same. That is, the partitioning of the frame is kept throughout the remainder of the trajectory and regular block matching is used to track the blocks instead of VSBM. To chain together the outputted displacements into a trajectory, the target block of a previous match is used as reference for the following matching. Also, regardless of the size of the trajectory, every frame serves as a starting point for a new set of trajectories. This is illustrated in Figure 4.2, where two distinct trajectories are calculated from the different partitions of two adjacent frames.

The experiments of Figueiredo (2015) provide some evidence that it is not necessary to use VSBM for every matching, since the results show no substantial improvement for using it so. Not only that, but using VSBM in every matching would cause the blocks to diverge and thus introduce more noise instead of relevant motion information. Also, very frequently the blocks would reach the minimum block size, since there was no merge operation. This would cause the method to lose one of its main characteristics: fast computation, since VSBM can be more demanding in the extreme case of a high number of small blocks per frame. By adopting VSBM for only the first pair of frames of the trajectory and block matching (BM) onwards, the speed of the overall procedure does not take a severe toll, while benefiting from a block configuration for the trajectory that approximates the boundaries of moving elements in the scene.

The reasoning behind this trajectory overlap scheme is that redundancy can increase the reliability of motion information, by diminishing the prevalence of noise. Should an object enter or exit the frame mid-trajectory, or if the sequence were to have distinctive

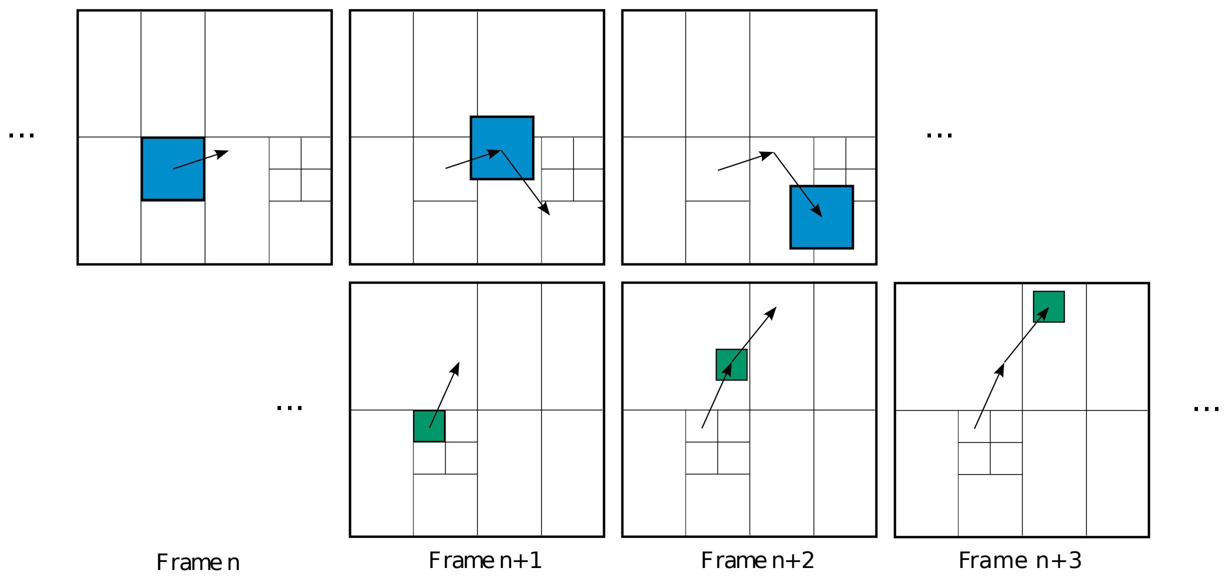


Figure 4.2: Example trajectories with size $S = 3$, starting at frames $f^{(n)}$ (top) and $f^{(n+1)}$ (bottom). On the top sequence, an object is tracked from $f^{(n)}$ to $f^{(n+2)}$ with the same partition. On the lower sequence, another block, originated from a distinct partition, is tracked from $f^{(n+1)}$ to $f^{(n+3)}$. In this case, the partition is the one resulting from the match between $f^{(n+1)}$ and $f^{(n+2)}$.

shots or cuts, the trajectories would be very likely to carry erroneous motion information, since the assumption of continuous motion would be toppled. By partially overlapping multiple trajectories, these errors have less impact on the overall accumulated motion information. Additionally, this redundancy scheme reinforces the idea that VSBM is not necessary for all the matchings. In the works of Wang et al. (2011), Wang et al. (2013a), and Caetano (2015), the trajectories have no overlap. In fact, in their methods, new interest points to spawn trajectories are extracted at each frame, but if a new point matches another point already in a trajectory, it is not taken into account. This essentially filters out trajectories with any overlapping.

Also differently from the dense trajectories of Wang et al. (2011), and Caetano (2015), the trajectories extracted through VSBM are sparse, in the sense that there are fewer trajectories per frame. This means a coarser representation of the motion in the scene compared to dense or keypoint sampling, but this smaller set of trajectories still possibly carries interesting motion information. Our evaluation of the method provides evidence towards this, as we achieve results that are marginally below those of the state-of-the-art in the area in experiments with the KTH dataset.

The extraction of the trajectories introduces some parameters and implementation decisions to our method. From VSBM, they are: initial block size (width/height in pixels), minimum block size, search strategy and error function used, error threshold, as well as whether to split blocks into two or four new ones. From the trajectory itself, we have its length (in frames). Some of these we have managed to identify as having very little to no impact at all in our results through preliminary experiments. And so for these parameters or aspects of the algorithm, we made choices envisaging simplicity, by way of ease and clarity of implementation, and fast computation.

Whenever the blocks are split, two new ones are created. As for minimum block size, we chose 4×4, to obtain a motion representation at most as dense as the largest scale of Wang et al. (2011) and Caetano (2015), where feature points are sampled on a grid spaced by 5 pixels[2]. For the search strategy and error function, we chose 4SS (PO; MA, 1996) and SAD, respectively, as previously stated.

4.4 HISTOGRAM DESCRIPTOR

Once the trajectories have been computed, we proceed to computing the descriptor. Even when employing sparse trajectories, the raw trajectories are still to much data to store directly as a descriptor. So a histogram of directions is used to summarize the motion information.

First, to clarify the notation, consider the indexes: $n \in [1, N]$ for the frame, $t \in [0, S]$ for the displacement vector in the trajectory, and $m \in [1, M_n]$, for the number of blocks in a partitioned frame.

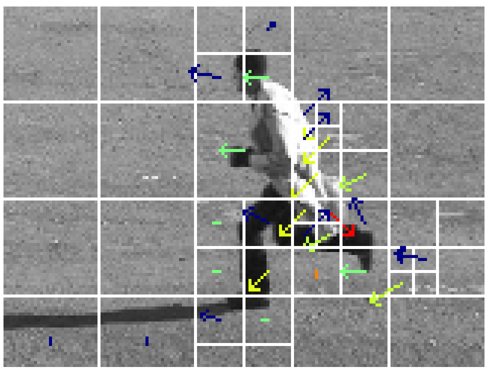
The input for these histograms are displacement vectors that compose the trajectories, and for that the vectors are first converted to equivalent polar coordinates. Let $\mathbf{d}_m^{(n+t)} = (d_x, d_y)_m^{(n+t)}$ be the t^{th} displacement vector of the trajectory associated with the m^{th} block of a given frame n through one of the matchings. In polar coordinates, $\mathbf{d}_m^{(n+t)}$ is denoted as $\mathbf{d}_m^{(n+t)} = (\theta, r)_m^{(n+t)}$, where $\theta = \tan^{-1}(\frac{d_y}{d_x})$, $\theta \in [0, 2\pi]$ and $r = \|\mathbf{d}_m^{(n+t)}\|$.

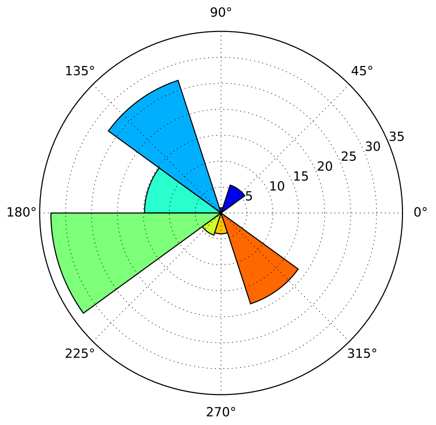
The histogram $\mathbf{h}^{(n)}$ for frame n is then defined as $\mathbf{h}^{(n)} = (h_1, h_2, ..., h_{P_\theta})^\top$, where P_θ is the number of bins for the θ coordinate of the vectors. The subdivision of the angle interval is uniform, that is, these bins are evenly spaced. The magnitudes for each bin are given by:

$$h_p^{(n)} = \sum_{m,t}^{M,S} r_m^{(n+t)} \cdot \omega(\theta), \tag{4.1}$$

[2]Still, a situation in which the whole frame would be partitioned into 4×4 blocks is highly unlikely to occur, considering the characteristics of natural videos.

where $p \in [1, P_\theta]$, and $\omega(\theta)$ is a Gaussian weighing factor, that adds part of a vector magnitude into neighboring bins. The closer to a interval boundary, the greater the contribution to the neighboring bin. In our experiments, this Gaussian factor has mean $\mu = 0$ and standard deviation $\sigma = 0.01$, meaning that only vectors very close to bin boundaries in terms of direction, contribute to more than one bin. Note, through the indexes of the summation, that all the displacement vectors that compose trajectories starting in frame n are used to build the n^{th} histogram of directions $\mathbf{h}^{(n)}$. The number of bins P_θ is a parameter of our method.


(a) Motion vectors/displacements.



(b) Corresponding histogram of directions.

Figure 4.3: Frame with motion vectors and its corresponding histogram of directions. To avoid clutter, only one motion vector per block is shown, and not entire trajectories.

4.5 TENSOR REPRESENTATION

A histogram could be used directly as the final descriptor, by taking the average or the concatenation of all frame histograms. Or it could, for example, be used in a bag-of-words method like in Wang et al. (2013a). However, a tensor representation is an interesting way to summarize motion information while easily retaining relations of similarity. And for that reason, we compute orientation tensors from our histogram motion descriptors as explained below.

Once we have histograms for all the N frames of a video sequence, we summarize them once again into a single final descriptor for the video sequence. For that, we first compute a tensor $\mathbf{T} \in \mathbb{R}^{P_\theta \times P_\theta}$ for each frame, defined as:

$$\mathbf{T}^{(n)} = \mathbf{h}^{(n)} \cdot \mathbf{h}^{(n)\top}. \tag{4.2}$$

Individually, $\mathbf{T}^{(n)}$ holds the same information as $\mathbf{h}^{(n)}$, but tensors can be combined to highlight component covariances. This combination, in our case, is an averaging of the tensors for all frames of a video sequence. The final descriptor for a given video sequence is then defined as:

$$\mathbf{T} = \sum_{n}^{N} \frac{\mathbf{T}^{(n)}}{\|\mathbf{T}^{(n)}\|} \tag{4.3}$$

where the norm $\|\mathbf{T}\|$ is the Frobenius L_2 norm, as cited previously in Section 3.2. By normalizing the tensors, we are able to compare sequences of different lengths and resolutions. Longer sequences or ones with higher resolutions would tend to have corresponding histograms with bins containing higher accumulated values, even if there was much less overall movement in said sequences. The normalization counteracts that and brings all the information into the same scale for comparison.

Two other properties of tensors are of great importance for our descriptor, as we want to have as much information from the trajectories as possible in a concise form. One is that, while vectors are representative of only one direction (that of the vector itself), tensors hold information of many directions. This essentially justifies the use of this tensor representation instead of using of just histograms as the final descriptor. In order to have available the information of all the histograms in their own form, we would have to concatenate them, as they are able to represent only one direction in $\mathbb{R}^{P_\theta}$. As with an orientation tensor in $\mathbb{R}^{P_\theta \times P_\theta}$ we have all N histograms from a sequence as line elements in $\mathbb{R}^{P_\theta}$ mapped into a surface in $\mathbb{R}^{P_\theta \times P_\theta}$. Figure 4.4 shows an illustration of three-dimensional second order tensors and how their shape represent the relation between directions encoded within them (KINDLMANN, 2004).

Note also that there is no increase in dimensionality between a tensor for a single frame and a tensor for all frames of a sequence. Both $\mathbf{T}^{(n)}$ and $\mathbf{T}$ are in $\mathbb{R}^{P_\theta \times P_\theta}$. Although this comes with a drawback that is brought up by Mota et al. (2013). If too much divergent information is accumulated on the tensor, it becomes isotropic (the one on the top-most part of Figure 4.4) and thus it does not hold meaningful motion information.

The other property of tensors of importance for this work is that the tensor has the

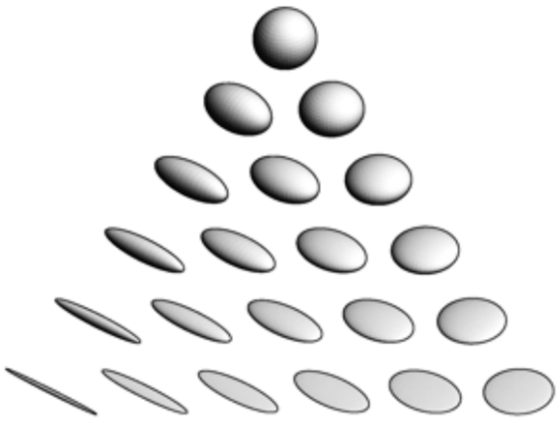


Figure 4.4: Illustration of three-dimensional second order tensors. Tensors with one prevalent direction on the bottom left, two prevalent directions on the bottom right, and all three (or without) prevalent directions on the top (KINDLMANN, 2004).

form of a positive semi-definite matrix. This means that in order to store it, only its upper triangular coefficients are needed. So instead of storing the whole matrix, with $P_\theta \times P_\theta$ coefficients, only $P_\theta \times (\frac{P_\theta+1}{2})$ coefficients are actually stored.

5 EVALUATION SETUP

This chapter contains the resources and some of the process behind our experiments. We evaluate our descriptors by their application in Human Action Recognition, as discussed in Section 5.1. Section 5.3 presents the datasets used in our evaluation, as well as some comments on their characteristics. Section 5.4 shows how we have explored the parameters of our method, for reproducibility's sake. Section 5.5 presents how we have adapted the Quadruplet-wise Metric Learning method into the context of our evaluation.

5.1 HUMAN ACTION RECOGNITION

Consider a set of video sequences containing actions being performed by people. The goal of Human Action Recognition (HAR) is to identify the actions or possibly the intents of one or more human agents in a scene. In fact, the problem consists in labeling these videos according to the actions portrayed. This topic attracts great interest due to its many potential applications. For instance, suppose an application to search or index videos automatically. In this application, the labeling could signal the presence of a certain kind of age-restricted or offensive content, or could differentiate between two genres, like sports or movies. A survey by Poppe (2010) covers an array of applications, developments, and challenges in this area.

In the context of this work, HAR serves as the means to evaluate our motion descriptors. That is, once we have a descriptor for each video sequence, carrying the motion information, we assign each video sequence to an action category according to its descriptor and those of sequences which the category membership is known. To make said assignments or predictions, very often a machine learning technique is considered, more specifically a supervised learning model of classification. The metric used to compare descriptors from different authors and somewhat measure the efficacy of the descriptors is the accuracy of the classification method, in the case of this work, that of a Support Vector Machine (SVM).

The classification is not an actual step of descriptor computation. It instead serves the purpose of making a distinction between kinds (categories, or classes) of descriptors. In HAR, these classes represent different actions. So, it is expected that the descriptors encode the characteristics of said actions into their coordinates. This should happen in a manner that it is possible to partition the high-dimensional space of the descriptors into regions that are populated by descriptors representing a single action category.

5.2 HARDWARE PLATFORM AND IMPLEMENTATION

All experiments were conducted either on a Intel®Core™ 2 Quad Q9550, 2.83GHz with 4GB memory or a Intel®Xeon™ E4610, 2.4GHz with 32GB. Any and all results concerning speed of computation were obtained using the Core™2 Quad computer running a single thread per video, with fairness of comparison in mind. The fact that only one thread was used serves to highlight how the speed of computation of the method could even be expanded through parallelism, since there is little to no data concurrency. More up-to-date hardware can increase the speed even further, although we expect that the benefits from it are much lesser than those that parallel computation can achieve.

Our software is written in both C/C++ and Python programming languages. And makes use of the OpenCV (ITSEEZ, 2015) and GCGLib [1] libraries.

5.3 DATASETS

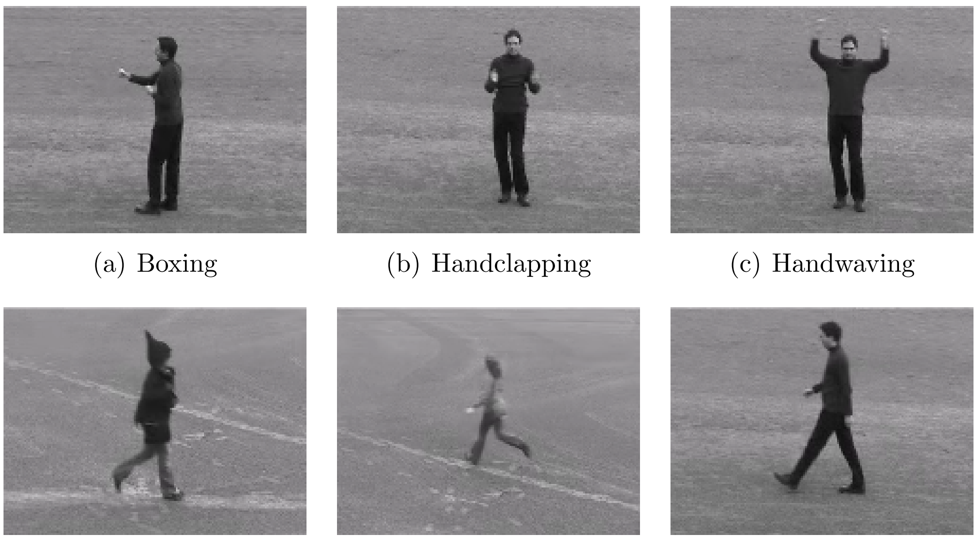
This section presents the datasets used in our evaluation. For all of them, we computed descriptors for each video in the dataset. These descriptors were then divided into training, validation, and testing sets for the SVM.

5.3.1 KTH

The KTH dataset (SCHULDT et al., 2004)(Figure 5.1) contains 600 videos of six human actions: *walking, running, jogging, boxing, hand waving and hand clapping.* These actions are performed by 25 people in four different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes, and indoors. As in (SCHULDT et al., 2004), the dataset is split in about four sequences for each video, producing a total of 2391 files. The sequences have a resolution of 160 $\times$ 120 pixels and 25 fps.

The sequences have a duration of about 2 to 3 seconds, are in gray-scale, and only one action is portrayed. No object or person other than the actor performing the intended

[1] GCGLib website: `http://www.gcg.ufjf.br/developments.html`



(a) Boxing (b) Handclapping (c) Handwaving

(d) Jogging (e) Running (f) Walking

Figure 5.1: Example videos from KTH dataset action categories (SCHULDT et al., 2004). Second row shows some of the different actors and camera viewpoints found in the database.

action appears in the same sequence. These properties are the reason this dataset is considered not very challenging or demanding. They are also the reason KTH is appropriate for our global self-descriptor, since all the movement in the scene is taken into account, without obstructions or any other kind of information.

5.3.2 UCF11 - YOUTUBE ACTION DATASET

The UCF11 dataset (LIU et al., 2009)(Figure 5.2) contains 1168 videos of eleven human actions: basketball shooting (b_shooting), volleyball spiking (v_spiking), trampoline jumping (t_jumping), soccer juggling (s_juggling), horseback riding (h_riding), cycling, diving, swinging, golf swinging (g_swinging), tennis swinging (t_swinging), and walking (with a dog). The sequences are organized into 25 somewhat independent groups, in different environments or made by different photographers. The sequences have low resolution, although higher than KTH dataset, and 29.97 fps.

In comparison with KTH, UCF11 is a much more complex dataset. It has shaky cameras, cluttered background, object scaling, varied viewpoints, and varied illumination conditions. These properties that UCF11 has, in themselves are enough to raise the level of challenge it poses for recognition. But the dataset also has a lot of similarity between some classes of actions. The classes: *b_shooting, v_spiking, t_jumping*, and *s_juggling* all

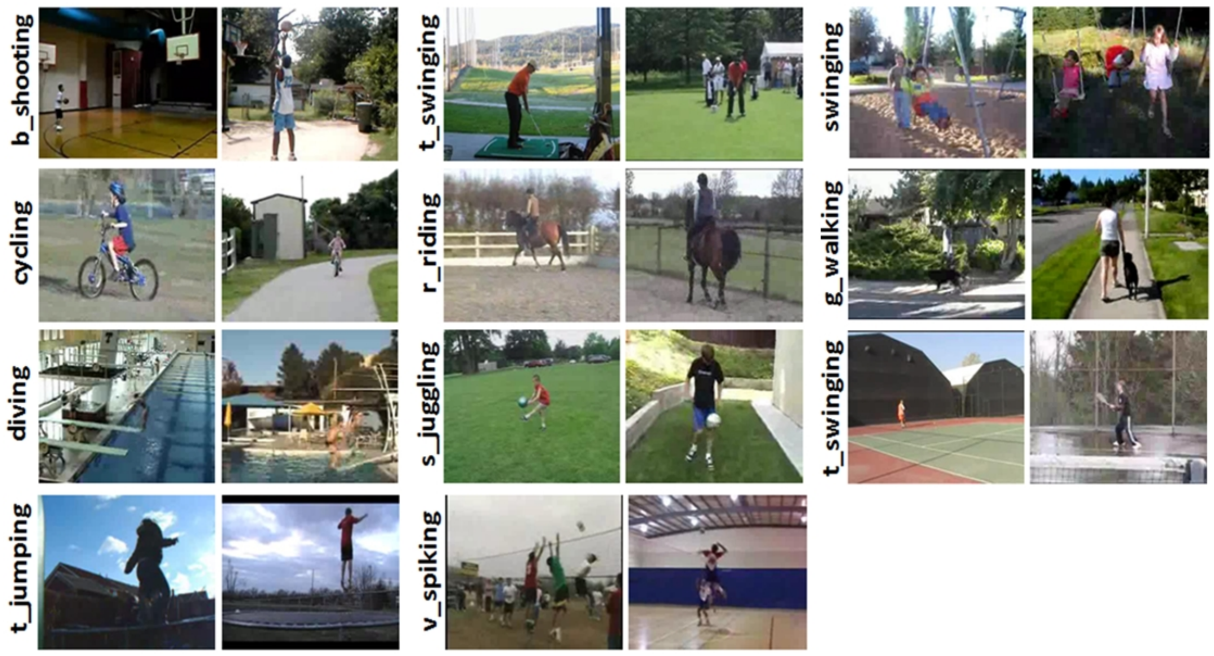


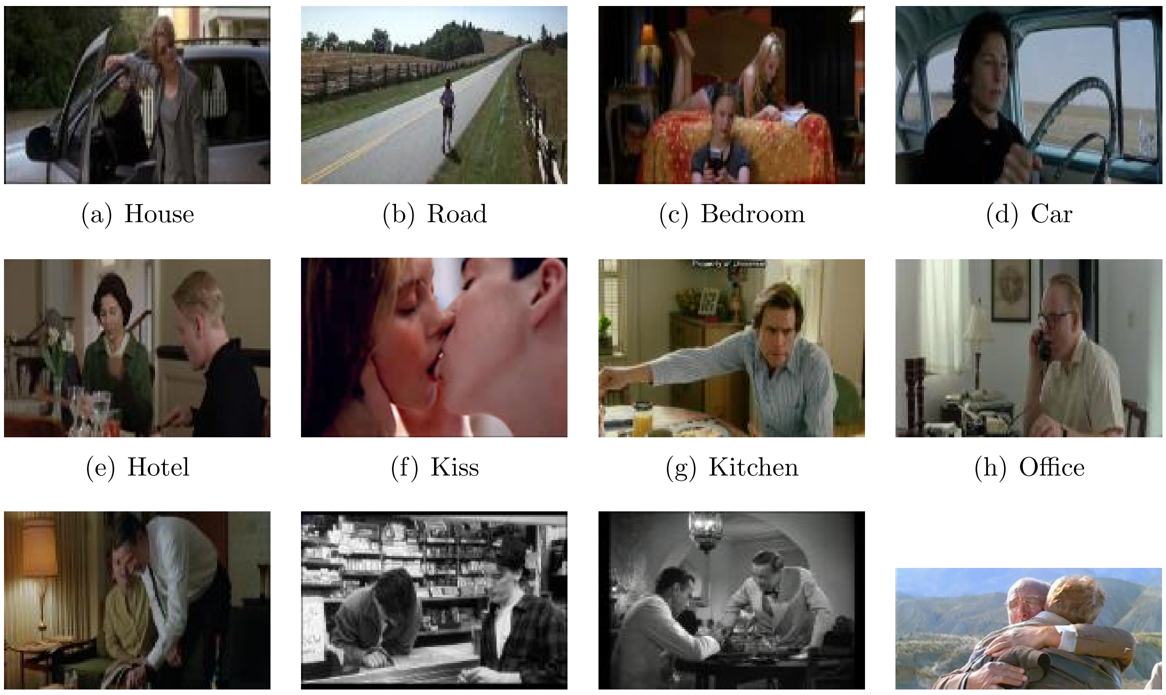
Figure 5.2: Example videos from UCF11 - YouTube Action dataset categories (LIU et al., 2009). Image from the authors' website: `http://crcv.ucf.edu/data/UCF_YouTube_Action.php`.

involve some form of jumping action, and there are three classes involving the swinging action. Finally, some of the actions, like *h_riding, cycling*, and *walking* are performed in conjunction with non-human actors, like horses, dogs, or bikes.

5.3.3 HOLLYWOOD2

The Hollywood2 dataset (MARSZALEK et al., 2009)(Figure 5.3) contains 3669 videos clips of twelve human actions: answer phone, drive car, eat, fight person, get out of car, handshake, hug person, kiss, run, sit down, sit up, and stand up. These clips come from 69 movie scenes, which are presented in medium to high resolutions and longer clip lengths than KTH or UCF11, to a total of approximately 20.1 hours of video. The list of movies, actions, and environments can be found at: `http://www.di.ens.fr/~laptev/actions/hollywood2/`.

Hollywood2 is undoubtedly the most challenging dataset of the three presented, in terms of Action Recognition. There is a very large variety of actors, backgrounds, environments, illumination, camera angles, as well as clip lengths. There are also some scene cuts and similar actions, like *sit up* can be confused with *stand up*, and *answer phone* with *eat*. The dataset has also a great deal of imbalance in number of videos per class.



(a) House (b) Road (c) Bedroom (d) Car

(e) Hotel (f) Kiss (g) Kitchen (h) Office

(i) Living Room (j) Shop (k) Restaurant (l) Hugging

Figure 5.3: Example videos from Hollywood dataset (MARSZALEK et al., 2009), showing a variety of actions and environments. Note the several camera angles, illumination conditions, backgrounds, and actors. Images and more detail at: `http://www.di.ens.fr/~laptev/actions/hollywood2/`.

The *sit up* class, for example, has a total of 51 video clips between training and testing sets, while the *run* class has 276 video clips.

5.4 PARAMETER EXPLORATION

Our method has several parameters, most of them from the extraction of trajectories. As previously stated in Section 4.3, they are: initial block size, minimum block size, search window size, search strategy and error function used, error threshold, block splitting factor, and trajectory size. From the descriptor computation, there are only two parameters: number of histogram bins and standard deviation for the Gaussian weighing factor.

Since there are so many parameters, exploring all of them thoroughly would be near unfeasible. So we attempted to identify those that are either secondary or that have less impact on the accuracy of the classification, and fixed their values. Some of these secondary parameters we have chosen values based on the available literature and from previous research. This is the case for the minimum block size and search window size, in accordance with that values used in the H.264 specification (MURALIDHAR C.B.

RAMA RAO, 2012). We have fixed other secondary parameters based on our own previous research and preliminary experiments, like the standard deviation and block splitting factor (VSBM tree arity). Table 5.1 shows the values for these parameters, for reproducibility's sake.

Table 5.1: Values for secondary parameters of our method, as well as some choices made for elements of the implementation, like the search strategy and error function.

Parameter	Values
Minimum block size	4×4 pixels
Search window size	15×15 pixels
Search strategy	Four step search (4SS)
Error function	Sum of absolute differences (SAD)
Block splitting	Binary (one block split in two)
Gaussian weight standard deviation	0.01

So the focus of our parameter exploration is on the initial block sizes[2], threshold values, trajectory sizes, and number of histogram bins. The ranges of values for our exploration of these primary parameters are all based on either the literature, like the initial block sizes and trajectory sizes, or on our previous experiments, like the threshold values. The only parameter we had to somewhat blindly estimate was the number of histogram bins, but even so we were oriented by the idea that the fewer the histogram bins, the more compact would the final descriptor be. Table 5.2 shows the values used for the experiments using the KTH dataset.

There are some results presented where the number of bins is 26, which might seem out of the pattern for experiments with varied histogram bins presented in Table 5.2. This is because these results come mainly from a previously published work of ours (OLIVEIRA; VIEIRA, 2015), where we were not concerned with the histogram itself and had fixed the number of histogram bins according to preliminary experiments. For this dissertation, we conducted more experiments, considering more values for the histogram bins parameter. The experiments in this work are more exhaustive, but the inclusion of previous results was necessary because the results were amongst the best ones. They are shown in Table 6.1 on a future section.

Another aspect worthy of note is that there is a preponderance of the trajectory size, block size and threshold over the other parameters when it comes to running speed. More blocks tracked for longer periods greatly increase the cost of computation. This is to be

[2]Or just block sizes, for short.

Table 5.2: Values for primary parameters of our method used in experiments with the KTH dataset.

Parameter	Values
Block size	$16, 24, 32, 40, 48, 56, 64$
Error threshold	$2000, 4000, 8000, 16000, 32000$
Trajectory size (frames)	$6, 7, 8, 9, 10, 11, 12$
Histogram bins	$10, 15, 20, 25, 26, 30, 35, 40$

expected as the complexity of the method lies much more on the number of matchings than on the size of the regions where SAD is computed. And in fact it is confirmed through our running speed experiments in Section 6.1.

Due to the increased size and resolution of both UCF11 and Hollywood2, we approach the parameter exploration for them in a slightly different manner. For UCF11 we chose a subset of the parameters used in KTH experiments, that contained the parameter combination corresponding to the best result, as shown in Table 5.3. For Hollywood2 we selected the same parameter values that yielded the best results for UCF11.

Table 5.3: Values for primary parameters of our method used in experiments with the UCF11 dataset.

Parameter	Values
Initial block size	$32, 40, 48, 56, 64$
Error threshold	$8000, 16000$
Trajectory size (frames)	$6, 7, 8, 9, 10$
Histogram bins	$10, 15, 20, 25, 30, 35, 40$

Note that so far, we are not taking into consideration the parameters of the SVM used to classify the descriptors. Table 5.4 contains these parameters and their respective values. The parameter σ affects how each feature from the samples for the SVM vary, between smoothly and abruptly. The parameter C is the regularization factor for the objective function of the classification. Both are responsible to control the trade-off between bias and variance of the SVM. These parameters are internally optimized by the SVM itself, through a validation step. So it outputs a total of 6 results, corresponding to the combination of kernels and power normalization exponents. From these 6 results, we choose the ones with highest prediction rate.

Table 5.4: Values for the parameters of SVM used to classify descriptors generated with our method. These values are used in the training step and a single combination of them is chosen during validation to be used in the test step.

Parameter	Values
Kernel	Triangular, Gaussian
p-norm power	$0.3, 0.5, 1.0$
σ	$0.2, 0.4, 0.6, 0.8, 1.0$
C	$1, 10, 100, 1000$

5.5 Q-WISE METRIC LEARNING ADAPTATION

For the sake of enrichment of our research, instead of just straightforwardly classifying the motion descriptors, we also propose an adaptation of the quadruplet-wise formulation for metric learning from Law et al. (2013), aiming to improve both the compactness and separability of the descriptors. Being based on the work of Parikh and Grauman (2011), the quadruplet-wise metric learning makes use of a table which associates attributes with relative ordering of classes, according to the presence of the attributes in each class. An example of said table (5.5) is provided below, as it is used for the OSR dataset in the work of Parikh and Grauman (2011).

Table 5.5: Relative orderings used for OSR dataset in both the relative attributes and quadruplet-wise metric learning. Classes: coast (C), forest (F), highway (H), inside-city (I), mountain (M), open-country (O), street (S) and tall-building (T).

OSR Attributes	Relative Ordering of Classes
Natural	$T \prec I \sim S \prec H \prec C \sim O \sim M \sim F$
Open	$T \prec F \prec I \sim S \prec M \prec H \sim C \sim O$
Perspective	$O \prec C \prec M \sim F \prec H \prec I \prec S \prec T$
Large-Objects	$F \prec O \prec M \prec I \sim S \prec H \sim C \prec T$
Diagonal-Plane	$F \prec O \prec M \prec C \prec I \sim S \prec H \prec T$
Close-Depth	$C \prec M \prec O \prec T \sim I \sim S \sim H \sim F$

The adaptation proposed for this work consists in making associations between classes, by their similarity amongst each other. For instance, *running* and *walking* are similar actions, while *running* and *jumping* are less so. Thus it is expected that as far as *running* is concerned, *walking* have a higher precedence than *jumping*, and the table would read "*jumping* $\prec$ ... $\prec$ *walking*" in the line where *running* takes the place of an attribute. The tables for the datasets used in this evaluation are Table 5.6, containing relative attributes for KTH, and Table 5.7, containing relative attributes for UCF11.

The Q-wise Metric Learning method introduces another parameter: the number of con-

Table 5.6: Adapted relative orderings used for KTH dataset. Classes: Boxing (B), Hand clapping (Hc), Hand waving (Hw), Jogging (J), Running (R), Walking (W).

Classes	Relative Ordering of Classes
Boxing	$R \sim J \prec W \prec Hw \sim Hc \prec B$
Hand clapping	$R \sim J \sim W \prec B \prec Hw \prec Hc$
Hand waving	$R \sim J \sim W \prec B \prec Hc \prec Hw$
Jogging	$B \sim Hc \sim Hw \prec W \prec R \prec J$
Running	$B \sim Hc \sim Hw \prec W \prec J \prec R$
Walking	$B \sim Hc \sim Hw \prec R \sim J \prec W$

Table 5.7: Adapted relative orderings used for UCF11 dataset. This table is summarized, and only the first classes in the precedence are presented. All absent classes in the ordering are assumed similar or equivalent for this matter. Classes: Biking (B), Diving (D), Golfing (G), Juggling (Jg), Jumping (Jp), Riding (R), Shooting (Sh), Spiking (Sp), Swinging (Sw), Tennis (T), Walk dog (W).

Classes	Relative Ordering of Classes
Biking	$\sim ... \prec W \prec R \prec B$
Diving	$\sim ... \prec Jp \prec D$
Golfing	$\sim ... \prec Jg \prec T \prec G$
Juggling	$\sim ... \prec G \prec B \prec Jg$
Jumping	$\sim ... \prec D \prec Sp \prec Sh \prec Jp$
Riding	$\sim ... \prec Sw \prec W \prec B \prec R$
Shooting	$\sim ... \prec D \prec Sw \prec Jp \prec Sp \prec Sh$
Spiking	$\sim ... \prec D \prec Jp \prec Sh \prec Sp$
Swinging	$\sim ... \prec D \prec Jp \prec Sw$
Tennis	$\sim ... \prec G \prec T$
Walk dog	$\sim ... \prec B \prec R \prec W$

straints derived from the relative orderings tables. In our experiments, we used $50, 100, 500$ and 1000 constraints, all built from random sampling of quadruplets out of the training set.

6 RESULTS AND DISCUSSION

This chapter presents the results the experiments with our method. First, in Section 6.1, the experiments with the method as described in the Proposed Method (Chapter 4), classifying computed descriptors straightforwardly. Next, in Section 6.2, the complementary experiments, considering the Quadruplet-wise Metric Learning approach to the classification.

6.1 SPARSE TRAJECTORIES

In this section, we present the results for each dataset in three different subsections. These results were obtained by computing and classifying the descriptors, without using the Quadruplet-wise Metric Learning method.

6.1.1 KTH

Table 6.1 shows the highest recognition rates and corresponding parameter values obtained during experiments on the KTH dataset. The case highlighted in bold face will be brought up for closer examination, using the confusion matrix from its classification (Table 6.2). The little variety in recognition rates indicates that the method is somewhat stable, that is, it performs in a similar manner regardless of its parameter values. Still, we question whether this stable behavior is inherent to the method or if there is need for even further exploration. There is also the fact that we select the best result out of the ones presented by multiple classifiers, which could be masking a very different landscape of results.

Figure 6.1 provides a more overall view of the same results through a contour plot. The results are the same published in Oliveira and Vieira (2015). Lighter colors indicate higher accuracy values. The highest value is marked as a black dot.

Both Table 6.1 and Figure 6.1 indicate that there seems to be no clear correlation between the parameters from the generation of the descriptor and the final recognition rate within the range of parameters explored, although cases with larger initial blocks tend to include the same information present on cases with smaller initial blocks, through the partition process. Clear exceptions are the cases with block sizes of 56×56 pixels, which clearly produces worse results. As discussed in Oliveira and Vieira (2015), this is

Table 6.1: Highest recognition rates obtained for the KTH dataset using the sparse trajectories histogram descriptors.

Block Size	Threshold	Trajectory Size	Histogram bins	Accuracy
24	2000	10	26	91.7
24	2000	11	26	92.1
40	4000	8	20	91.7
40	4000	9	30	91.2
40	4000	10	25	91.7
40	**8000**	**6**	**26**	**92.1**
40	8000	8	25	91.7
40	8000	12	30	91.7
48	8000	9	26	92.1
48	8000	10	26	92.1
48	8000	11	26	91.7
48	8000	12	26	91.7

due to the ratio between the block size and dataset resolution. KTH has a resolution of 160×120 pixels, meaning that only two 56×56 sized blocks fit vertically on a frame and a third block is composed of 48 rows of out-of-bounds pixels. Our method, as explained in the same paper, is implemented with a border/out-of-bounds treatment to minimize undesirable scenarios. But regardless, some cases like this one are bound to happen, even more so in datasets with varied resolutions, and thus introduce noise into the descriptor along with the desired motion information.

Table 6.2: Confusion matrix of the best result in KTH dataset. The average recognition rate is 92.1% in this case. Columns indicate the predicted class, while rows indicate the actual class.

	Boxing	Clapping	Waving	Jogging	Running	Walking
Boxing	97.2	2.8	0.0	0.0	0.0	0.0
Clapping	11.1	88.9	0.0	0.0	0.0	0.0
Waving	2.8	0.0	97.2	0.0	0.0	0.0
Jogging	0.0	0.0	0.0	94.4	2.8	2.8
Running	0.0	0.0	0.0	25.0	75.0	0.0
Walking	0.0	0.0	0.0	0.0	0.0	100.0

Table 6.2 shows the confusion matrix corresponding to the best result in KTH, regarding recognition rate. It is easily observable that the matrix is somewhat divided into two square blocks. One showing the confusion between *boxing*, *hand clapping*, and *waving*. The other showing the confusion between *jogging*, *running*, and *walking*. This is to be expected due to the nature of the scenes. On the first three action classes, the actors are standing in the middle of the frames, moving only their arms in repetitive patterns. On

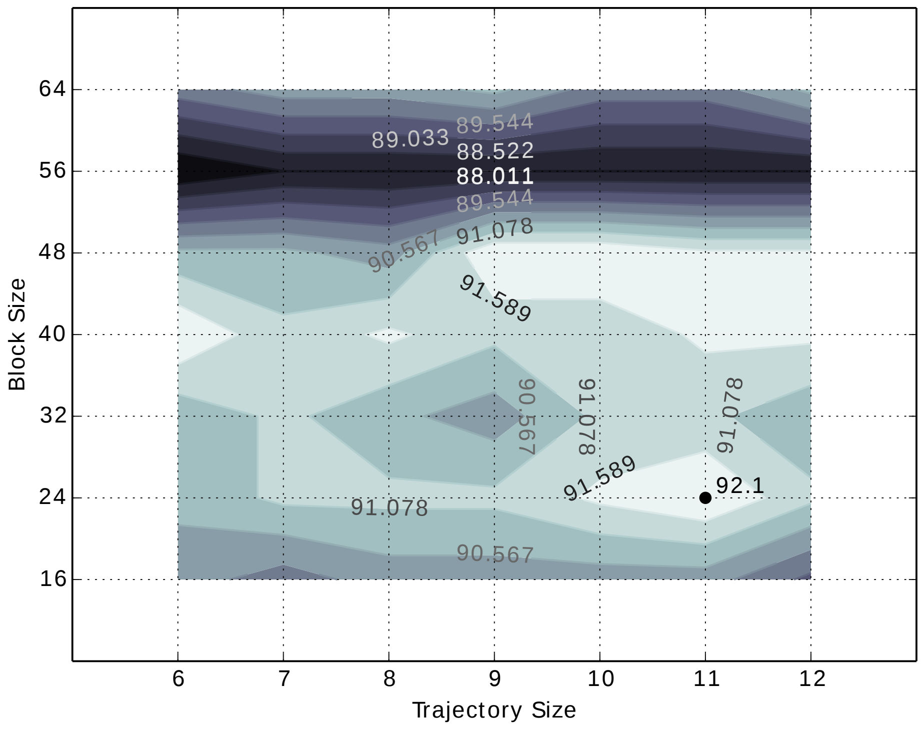


Figure 6.1: Contour plot showing classifier accuracy/recognition rate for each parameter setting regarding block and trajectory sizes in KTH experiments. Lighter colors indicate higher accuracy values. The highest value is marked as a black dot. Already published in Oliveira and Vieira (2015).

the last three action classes, the actors move around, and often leave, the frame.

The confusion between *boxing* and *clapping* is due to the repetitive movement of the actor's arms nearby his body. *Waving*, however, is characterized by more expansive movements of the actor's arms, hence its lesser confusion with the other two.

We consider that the main responsible for the high confusion between jogging and running to be the subtle difference in rhythm. In terms of the actual movement performed, both are essentially the same, except for the speed in which the action is performed. This difference is not properly captured by our method, since the tensors are normalized with the intention of making possible to compare scenes of different lengths and resolutions.

Note how this confusion matrix analysis reinforces the idea of visual similarity between kinds of movements. Similar movements are in fact represented by tensors nearby each other in a high-dimensional space. So much so that the SVM is not capable of perfectly separating these tensors. We consider this evidence that our descriptors, the way they are computed, encode these relations of similarity appropriately.

Moreover, the recognition rates for KTH also show little sensitivity to variations in the parameter combinations. The lowest points in Figure 6.1 are recognition rates of around 88%, while the highest around 92%. This is a small difference, considering the somewhat extensive parameter exploration in the case of this dataset.

This is not unfortunate though, as we regard this is evidence for the limits of our method over simple datasets, like KTH. Take into consideration that our method is a very simple, global approach, and that the dataset is simple enough to have results with at least a modest amount of insensitivity to changes in the parameter settings. In this scenario, instead of focusing on reaching even higher recognition rates, we can explore the impact of the parameter settings on other aspects of the method, like running speed and final descriptor size.

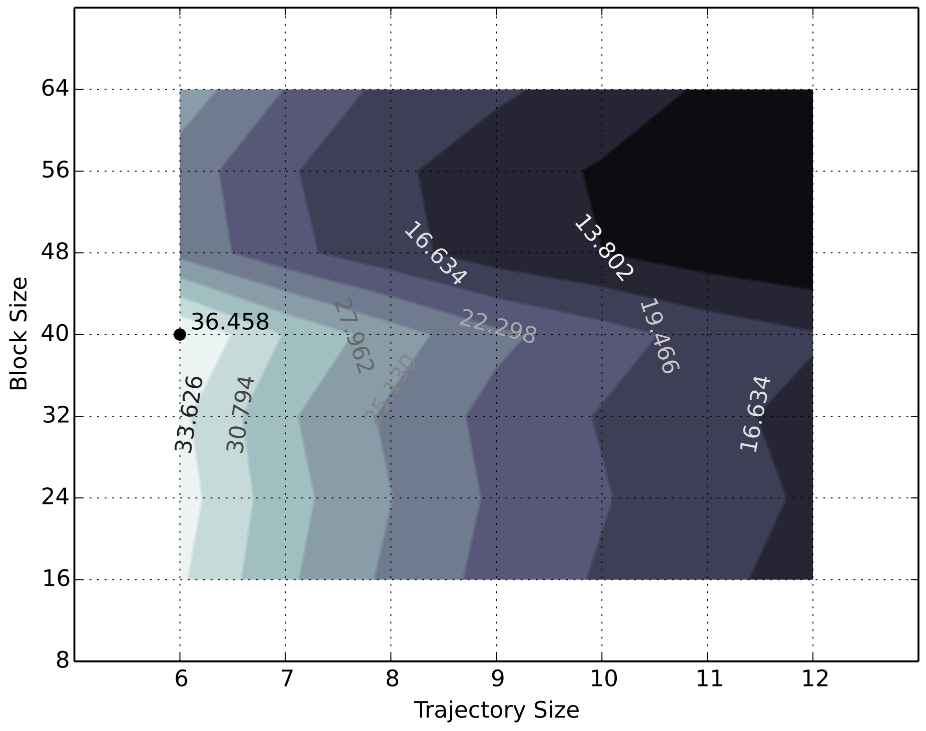


Figure 6.2: Contour plot showing running speed (in frames per second) for each parameter setting regarding block and trajectory sizes in KTH experiments. Lighter colors indicate higher frame rates. The highest value is marked as a black dot. Already published in Oliveira and Vieira (2015).

Figure 6.2 shows an overview of the speed of computation of our method. The values depicted are the mean frame rates for KTH videos. To obtain these values, we measured the time taken to compute the block matching trajectories, histogram descriptor and tensor representation. No classification or any overhead due to file handling or external libraries[1] were taken into account for this measurement. With this running time, we computed the frame rates for each sequence and then the mean considering all sequences. Considering the simplicity and uniformity of KTH samples, this mean is a good representation of the actual running speeds for each sequence individually.

Note that the trajectory size has a major impact on the frame rates, as it was expected. The greater the trajectories, the more matchings computed in a set of frames. Also as expected, the block size only has a minor impact in this sense, along with the threshold, whose influence is not visible in Figure 6.2. The bigger the initial blocks, the less matchings are computed. But in this case, if the threshold is low, more splitting operations are carried out and the number of matchings becomes very high.

To exemplify the benefit of the insensitivity of the recognition rates with respect to parameter variations, consider the lines of Table 6.1 highlighted in Table 6.3:

Table 6.3: Summarized KTH results.

Block Size	Threshold	Trajectory Size	Histogram bins	Accuracy	Frame rate
24	2000	11	26	92.1	16.7
40	8000	6	26	92.1	36.4
40	8000	12	30	91.7	16.5

These combinations of parameters all yield similar recognition rates, but the difference in performance of the method with each of these combinations is very noticeable. The frame rate for the case highlighted on the second line of Table 6.3 is the highest one amongst the experiments, approximately 36 fps, while the frame rate for the cases on the first and third lines are below half this value. This indicates that, in an application scenario, if there were a choice to be made between these parameter combinations, there would be absolutely no need for the computation of trajectories of size 12. Moreover, considering that videos are usually captured and played at a speed of approximately 24 fps, we can affirm that our method can be used in tandem with the capture of images and still provide the best results it can achieve.

Just as a basis for comparison of the speed of computation between our method and some other common approaches, consider the method by Wang et al. (2011). Despite being distinct in essence from our method, by the use of local features along their trajectories, their work serves as a base for many subsequent works. In their method, feature points are

[1]In this case, the library used to manipulate video files was OpenCV (ITSEEZ, 2015).

densely sampled from the dense optical flow of Farnebäck (2003) in multiple spatial scales. Then each point is tracked throughout the trajectories and has HOG, HOF and MBH computed in its vicinity. Despite not having any overlap of trajectories, the computation of the tracking, the optical flow and the feature point sampling in multiple scales is required for every pair of frames. Note that this is an oversimplification of their framework, since after these cited procedures, their descriptor is not yet computed. These local features extracted undergo bag-of-words encoding before finally composing the descriptor for each sequence. This simplification benefits their method for the sake of the speed comparison intended, but it is enough to provide an upper bound for the speed of computation of dense trajectories, as any other computation required are bound to slow the method even further.

The computation of HOG implemented in OpenCV (ITSEEZ, 2015) runs at approximately 72 fps for KTH sequences. This might seem much faster than our method can perform, but this value is computed considering only one HOG computation per pair of frames, instead of one per feature point in a frame. As for the optical flow of Farnebäck (2003), the running speed of the implementation of it in OpenCV is of approximately 37 fps, once again considering a single computation per pair of frames, instead of one for every spatial scaling of a pair of frames. There is also the matching between local descriptors for every feature point to construct the trajectories and the computation of MBH, for which we have not found available code in OpenCV, but expect to add to the cost of computing their descriptors.

In contrast, our speed measurements were made considering the whole process of generating the descriptor, from the construction of trajectories to the computation of the tensor representation. So if anything, this comparison does not work in our benefit.

Regardless, the computation of the optical flow alone puts a limit to the speed of the dense trajectories method. Considering a gross simplification, we estimate that its upper bound is at most as fast as our method, that runs at between 13 and 36 with the parameter settings we experimented. Admittedly, a more thorough analysis of the performance of other methods as a whole would be much more preferable. But these values provide at least some perspective, and hint at a baseline performance for any method which makes use of HOG, HOF, and dense trajectories.

6.1.2 UCF11

Table 6.4 presents the highest recognition rates and corresponding parameter values obtained during experiments on the UCF11 dataset.

Table 6.4: Highest recognition rates obtained for the UCF11 dataset using the sparse trajectories histogram descriptors.

Block Size	Threshold	Trajectory Size	Histogram bins	Accuracy
32	8000	6	26	63.8
32	**8000**	**7**	**26**	**64.8**
32	8000	8	30	64.1
32	8000	9	30	64.1
32	8000	9	26	63.4
32	8000	10	30	64.0
32	16000	6	25	63.9
32	16000	10	26	63.4
56	8000	7	26	63.3
56	8000	8	26	63.3
56	8000	9	26	63.6
56	8000	10	26	63.3

Figure 6.3, like the previous figure, provides a more overall view of the same results through a contour plot. Lighter colors indicate higher accuracy values. The highest value is marked as a black dot.

Note how the recognition rates seem to have a greater sensitivity to changes in the parameter combinations than in KTH results. Not only the best results are much more diverse, in terms of recognition rates, but some parameter combinations yielded as low as 58% recognition rates. Additionally, amongst the 12 best results presented in Table 6.4, there are more combinations of parameters than in KTH results (Table 6.1), even with the more limited parameter exploration.

Albeit appearing to have a great sensitivity on UCF11, the method can still be considered somewhat stable. Again, the stability or insensitivity could be product of the selection of the best results amongst the classifier outputs. Using UCF11, experiments with a wider gamut of parameters could more easily reveal different behaviors of the method. As discussed in the KTH results section, this is positive in the sense that more combinations of parameters can be used to improve other aspects of the method without necessarily hurting its recognition capacity. But more experiments are absolutely required to investigate whether the stability is really a property of the method or just a particular

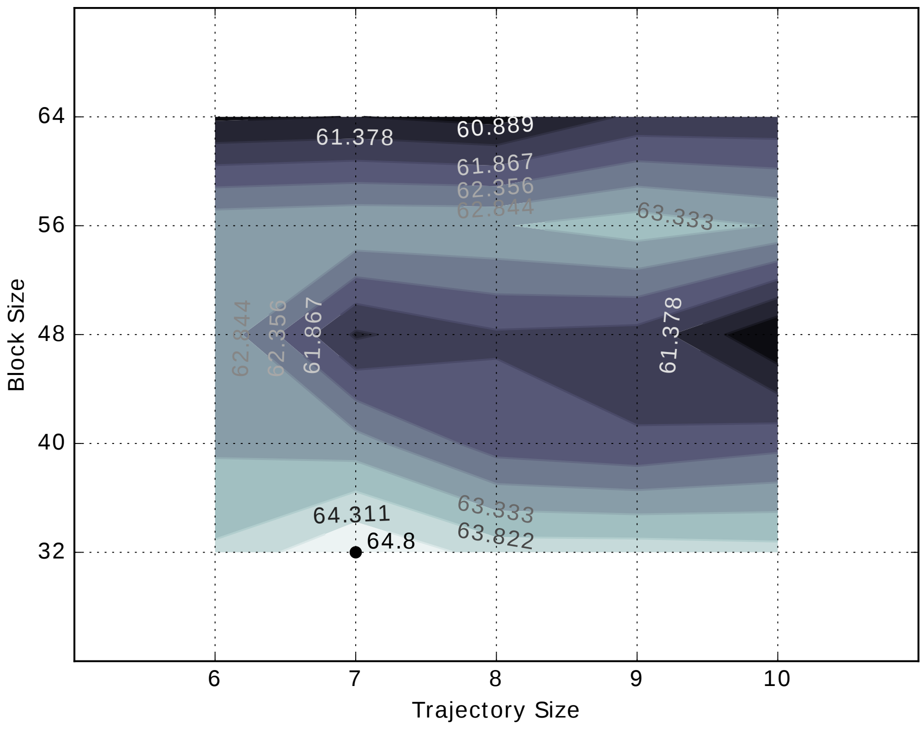


Figure 6.3: Contour plot showing classifier accuracy/recognition rate for each parameter setting regarding block and trajectory sizes in UCF11 experiments. Lighter colors indicate higher accuracy values. The highest value is marked as a black dot.

set of parameters and circumstances that makes it seems so.

Table 6.5 shows the confusion matrix for the case with highest recognition rate on UCF11 dataset. It is noticeable how this matrix is much more dispersed than the one for KTH. There are more distinct action samples that are wrongfully classified, which is indicative of a greater agglomeration of the descriptors for said samples in the classification space.

The most prominent confusion is between actions of similar nature. *Bike*, *ride* and *walk dog*, for instance, are specially characterized by the interaction with objects, rather than by the movement of the actors themselves. Without previous knowledge of the presence of a bike, a horse, or a dog in the scene, it is hard to distinguish between samples of these actions. Another example is the confusion between *juggle* and *jump*, which have the same characteristic up and down movement, but in scenes from the *juggle* class, the ball being juggled is performing the movement, while in scenes from *jump*, it is a person jumping on a trampoline that actually performs the action. *Shoot* and *spike* are confused because they present similar actions, in which the actor jumps and a ball travels at a high speed

Table 6.5: Confusion matrix of the best result in UCF11 dataset. The average recognition rate is 64.8% in this case. Columns indicate the predicted class, while rows indicate the actual class.

	Bike	Dive	Golf	Juggle	Jump	Ride	Shoot	Spike	Swing	Tennis	WDog
Bike	62.4	0.7	0.0	0.7	0.0	13.3	2.7	0.0	1.5	6.3	12.5
Dive	1.6	77.6	1.6	3.6	1.2	4.0	3.2	1.6	0.6	4.4	0.7
Golf	0.0	1.7	78.1	6.4	0.0	1.6	4.4	0.7	1.0	5.6	0.6
Juggle	1.3	1.3	5.3	56.8	12.1	1.2	3.6	2.0	5.6	9.2	1.5
Jump	0.7	1.7	1.0	11.1	73.5	0.0	0.0	0.0	12.1	0.0	0.0
Ride	8.2	0.4	0.0	0.7	0.0	80.6	1.1	2.9	0.7	1.2	4.3
Shoot	2.8	9.8	3.1	7.5	0.8	2.0	48.7	12.4	0.7	8.9	3.2
Spike	1.7	9.6	0.0	3.0	1.8	0.7	10.8	60.0	0.0	7.3	5.1
Swing	3.9	0.8	1.0	4.1	13.8	0.0	0.0	0.0	74.5	1.3	0.5
Tennis	2.9	8.3	8.3	9.4	1.1	3.6	6.0	3.6	1.1	54.5	1.1
WDog	11.0	0.8	3.3	1.0	0.8	18.7	5.9	4.6	3.9	3.7	46.2

following the jump from the actor.

Note that this characterizes evidence that our descriptors properly and effectively encode visual similarity between distinct kinds of movement, even in more complex scenarios, such as the ones depicted in UCF11 scenes.

Concerning the running speeds, there are no special remarks to be made as results for UCF11 follow the same behavior as in KTH. Frame rates for UCF11 runs are lower, though, ranging from approximately 5 fps to 15 fps.

6.1.3 HOLLYWOOD2

For the Hollywood2 dataset, we experimented only with little variations around the parameter values that yielded the best results for the UCF11 dataset. The highest prediction accuracy for Hollywood2 achieved was 32.5%.

Table 6.6: Recognition rates obtained for the Hollywood2 dataset using the sparse trajectories histogram descriptor. The parameter combinations considered were only the few around the combination that produced the best results for UCF11.

Block Size	Threshold	Trajectory Size	Histogram bins	Accuracy
32	**8000**	**7**	**26**	**32.5**
32	16000	7	26	32.1
64	**8000**	**7**	**26**	**32.5**
64	16000	7	26	31.8

Table 6.6 presents the recognition rates and corresponding parameter values obtained during experiments on the Hollywood2 dataset. The recognition rates shown are much

lower than for the KTH and UCF11 datasets. As mentioned before, Hollywood2 is much more complex, so these recognition rates were to be expected.

Sensitivity to parameters is much harder to assess in Hollywood2 experiments, with such a narrow range of parameters to explore.

Table 6.7: Summary of the confusion matrix of the best result in Hollywood2 dataset. The average recognition rate is 32.5% in this case. The entries are the accuracies for each action category. The cross-class confusion is not shown for clarity's sake, since there are 12 action classes in total.

AnswerPhone	Drive	Eat	Fight	GetOutCar	Handshake
18.8	62.8	17.6	62.3	15.6	10.9
Hug	**Kiss**	**Run**	**SitDown**	**SitUp**	**StandUp**
18.6	37.7	51.9	45.5	8.0	38.9

Table 6.7 shows the confusion matrix for the case with highest recognition rate on Hollywood2 dataset. Although the whole confusion matrix is not presented, we can note how there are very challenging classes, especially the ones that involve the use of some object. In these classes, such as *answer phone*, *eat*, and *hug*, the movement of the actor himself is not very distinctive of the class. These actions are rather dependent on another actor, like *hug*, or on an object, such as a phone in *answer phone*. On top of that, these three actions are often captured in camera close-ups. So for most of the clips, only the bust or head of the actor is in frame, and the phone, food, or other actor come into frame very suddenly.

The classes with more fixed angles and very distinctive, characteristic movements, on the other hand, have much higher prediction accuracy. *Drive*, *fight*, *run* and *sit down* are examples of such classes.

6.1.4 STATE-OF-THE-ART

Table 6.8 provides a summary of results, both from the state-of-the-art and our method. We understand that there is an array of other works, even more recent and also with very high recognition rates. Still, we wanted to keep this comparison between approaches that are based on motion extraction and description, like our own method.

The first commentary to be made based on this table concerns the recognition rates for the KTH dataset. Albeit still below those from other works, our method achieves similar results. We attribute this approximation to the combination between the simple scenario

Table 6.8: Comparison between our method, related works, and state-of-the-art highest recognition rates for KTH, UCF11, and Hollywood2 datasets.

	KTH	**UCF11**	**Hollywood2**
Kläser et al. (2008)	91.0%	–	24.7%
Wang et al. (2011)	94.2%	84.2%	58.3%
Mota et al. (2012)	93.2%	72.7%	40.3%
Perez et al. (2012)	92.0%	–	34.0%
Guo et al. (2013)	98.5%	78.5%	
Mota et al. (2013)	92.5%	75.4%	40.3%
Jain et al. (2013)	–	–	62.5%
Wang et al. (2013b)	95.3%	89.9%	59.9%
Caetano (2015)	94.1%	–	46.5%
Figueiredo (2015)	91.4%	65.8%	41.5%
Our method	92.1%	64.8%	32.5%

that KTH provides and our global descriptor. In scenes with just a single action, no clutter or shot cuts, a global descriptor carries very little noise in relation to the motion information encoded. We consider this an evidence that simple, global descriptors are adequate for simple, controllable environments. A dense trajectory or dictionary based approach in these simple scenarios is computationally too expensive to justify very little gain in relation to our simpler method.

In such cases, we consider our method to be on par, if not advantageous due to its simplicity, in relation to those from other works. For instance, a direct application of our approach could be a surveillance system where the cameras are static and the background is very well-known and predictable. Another similar system, but without these characteristics, like a metro/subway surveillance system, would require a more sophisticated approach to handle the large amount of people moving in the scene. Then again, depending on how much more complex the method would need to be, the corresponding speed cost could make this application unfeasible.

This approximation does not hold for higher complexity scenes, however, as indicated by the larger gap between recognition rates for UCF11 and Hollywood2. As the complexity increases, the redeeming characteristics of our method become less advantageous, to the point of possibly being detrimental to the recognition rates. Our descriptor is global, meaning that it contains information from whole frames and whole sequences, without pre-segmentation of elements, feature extraction or prevalence of local spatio-temporal information. It is also what we call a self-descriptor, that is, a descriptor for a given sequence does not rely on information from other sequences.

In the more complex scenarios, a large amount of noise is introduced into the descriptor, due to background clutter, occlusions, unpredictable camera movement. On top of these visual challenges, the higher number of action categories and the disposition of the descriptors in the classification space make the distinction of classes much harder. Due to the higher variety of scenes in each action class, the descriptors are much more conglomerated and the classes share the same regions of the classification space. This is where bag-of-words or other dictionary approaches become advantageous. By constructing a predictable vocabulary of descriptors and thus alleviating the challenges introduced by the complexity of the scenes. We estimate that both our sparse trajectories and histogram features could be incorporated into such an approach successfully. This is out of the scope of this dissertation, but could be a future improvement.

So, to address our first two hypotheses, concerning whether trajectories of block matchings and histograms are able to produce a proper descriptor for motion, we regard that they have been confirmed through our experiments, as long as we add a restriction. The dataset, or domain of application is required to be under controlled conditions and be considerably simple, in order for our method to perform comparably to the state-of-the-art. Complex environments and sequences are beyond what our descriptor is capable of handling with proper efficacy. This is due to a combination of a limitation of tensors, and our global self-descriptor approach. That is, in complex scenes, the tensors have a higher susceptibility to becoming isotropic due to too much conflicting directions information, coming from different elements in the frame. Local approaches, more sophisticated dimensionality reduction, and dictionary-based methods are expected to handle these cases more appropriately. On top of that, if a scene involves a large amount of moving elements, the speed performance of our method is also diminished.

That is not to say that our two hypotheses mentioned are to be discarded, even without the restriction. They are also corroborated considering that under similar conditions to works involving both tensors and trajectories, like Caetano (2015) and Figueiredo (2015). These works use a combination between the trajectories and HOG using outer product. We achieve comparable results to theirs, in terms of recognition rates, but with a much more straightforward and fast-computing scheme, using only histograms of orientation from trajectories.

6.2 METRIC LEARNING

In this section, we present the results for the additional experiments we conducted, using an adaptation of Quadruplet-wise Metric Learning, as described in Section 5.5. Unless otherwise stated, the experiment conditions were the same as in our previous experiments, presented in Section 6.1. For these experiments, we computed the descriptors, followed by the metric matrix, which was used to transform the descriptors prior to their classification.

6.2.1 KTH

Table 6.9: Results using Q-wise Metric Learning for the classification of the highest recognition rate scenarios presented in Table 6.1.

Block Size	Threshold	Trajectory Size	Number of constraints	Accuracy
24	**2000**	**10**	**26**	**86.9**
24	2000	11	26	83.9
40	4000	8	20	83.4
40	4000	9	30	84.4
40	4000	10	25	78.9
40	8000	6	26	82.1
40	8000	8	25	69.8
40	8000	12	30	81.3
48	8000	9	26	73.0
48	8000	10	26	71.2
48	8000	11	26	71.5
48	8000	12	26	73.0

Table 6.9 shows the highest recognition rates and corresponding parameter values obtained during experiments on the KTH dataset using the Q-wise Metric Learning adaptation explained in Section 5.5. These recognition rates are also responses from an SVM under the same conditions as the previously reported results. This time though, the resulting matrix learned $\mathbf{L}$[2] is applied as a linear transformation to all the samples, obtaining a higher level feature vector representation of them before classification.

For all of these cases, the number of constraints was the highest tested, 1000 constraints. This signals that the more constraints the better, to properly learn the metric. More experiments are needed to find if there is an upper bound for this number before facing diminishing returns. Note, though, that the more constraints, the more time and computation it takes to learn $\mathbf{W}$.

[2]Refer back to Section 3.3 for more details on the matrix $\mathbf{L}$ and its role.

The results are worse in all experiments conducted, and the differences range from less than absolute 5% to 20%. This goes against our hypothesis of improved separability of the samples. However, the dimension of this new vector representation is the same as the number of relevant attributes, or in the case of our adaptation, the number of action classes. This means that the descriptors for KTH become *six-dimensional vectors* after this transformation. That is as little as 24 bytes of information per video, if the entries are single precision floating point numbers.

Further exploration of this method is highly recommended, since we have not taken into account different relative orderings, a more thorough parameter optimization, or a more hand-crafted generation of constraints. But we regard these results as being very positive, considering how they are indicative of a great capability of encoding the characteristics of each class. Furthermore, even though we cannot confirm our hypothesis of Metric Learning improving the separability of the samples, we do not discard it either, on the basis that our approach was just an early attempt at incorporating it into the action recognition problem.

6.2.2 UCF11

Table 6.10: Results using Q-wise Metric Learning for the classification of the highest recognition rate scenarios presented in Table 6.4.

Block Size	Threshold	Trajectory Size	Histogram bins	Accuracy
32	8000	6	26	45.2
32	**8000**	**7**	**26**	**46.2**
32	8000	8	30	41.3
32	8000	9	30	44.1
32	8000	9	26	39.1
32	8000	10	30	44.8
32	16000	6	25	42.8
32	16000	10	26	43.9
56	8000	7	26	40.4
56	8000	8	26	39.7
56	8000	9	26	40.5
56	8000	10	26	41.0

Like in experiments using KTH, the results shown in Table 6.10 are lower than their counterparts with just the SVM classification of our descriptors, presented in Table 6.4. Again, all of best results reported were yielded by cases with 1000 constraints, the highest value tested for this parameter.

Despite these similarities with results from KTH, this time the best result was not produced by the same descriptor that achieved the highest recognition rate in the previous results (Table 6.4). Moreover, through observations of Table 6.10 there seems to be no correlation between recognition rates with and without Metric Learning. Finding and identifying such a relation (or lack thereof) could be another point of analysis for future works.

Albeit being considerably below the recognition rates without the use of Metric Learning, with differences reaching upwards of absolute 21%, the descriptors are once again severely compacted, becoming just eleven-dimensional. In comparison, the descriptor that produced the best result without using Metric Learning has 351 floating point entries.

This specific approach does not provide an improvement, which might be indicative of a limitation of such compact descriptors. A more thorough investigation is required from the adaptation of the Metric Learning method to the parameter exploration, to achieve better results in the more complex scenarios of UCF11 and other datasets.

7 CONCLUSION

In this work, we proposed a histogram descriptor for motion in sequences of images, based on trajectories of block matching displacements. The use of trajectories based on block matching have been of good efficacy and efficiency, as we could account by the results obtained. The same can be said for condensing information using the histograms of relative displacements, since the results are comparable to those of more complicated methods, that make use of local information, like Caetano (2015) and Figueiredo (2015). These hypotheses can also be extended with the use of trajectories and histograms in dictionary-based approaches.

We validate our method using the recognition rates over three datasets, in a human action recognition application. For the KTH dataset, the best setups reach 92.1% prediction accuracy, with initial block sizes between 24×24 and 48×48 pixels, splitting thresholds between 2000 and 8000, trajectory sizes between 6 and 11 frames, and 26 histogram bins. For the UCF11 dataset, the best setups reach 64.8% prediction accuracy, with initial block sizes between 32×32 and 56×56 pixels, splitting thresholds between 8000 and 6000, trajectory sizes between 6 and 10 frames, and 26 histogram bins. For the Hollywood2 dataset, due to deadline constraints, we experimented only with the parameter values that yielded the best results for the UCF11 dataset. The prediction accuracy for Hollywood2 achieved was 32.5%.

The recognition rates in KTH, which contains simple scenarios, were comparable to those of much more complex methods involving even dictionary approaches and much denser trajectories and brightness flows. In the more complex datasets, UCF11 and Hollywood2, our recognition rates are considerably lower than those from the state-of-the-art methods. Thus leading us to the conclusion that a simple, global self-description approach like ours is much more applicable in a equally simple scenario. The background clutter, multiple cuts, multiple actions, and other difficulties posed especially by Hollywood2 ask for higher complexity solutions catered specifically to tackle these challenges.

The trade-off between compactness, speed, and accuracy seems to be a non-trivial matter. In fact, as far as can be seen through the results, it does not even characterize a trade-off. The compactness and speed of computation of the descriptors are much more predictable through the values of the parameters than the accuracy of the descriptors. Whereas for the accuracy, we could not properly assess the influence of the parameters in order to tune them for better results.

This does not mean that this work cannot be improved upon to achieve better results than the ones presented. A simple improvement, with no alterations to the method itself, could be to conduct a more thorough exploration of the parameter space, eliminating parameter confounding and redundancy, as well as properly assessing their impact on the overall scheme. Another improvement would be to aggregate tensors in a different manner than just averaging them, what is known to produce isotropic tensors, given certain conditions. In the work of Yuan et al. (2010), the authors benefit from the fact that orientation tensors are similar to covariance matrices, and thus do not lie on Euclidean space, but instead on a Log-Euclidean Riemannian metric space. The same metric could also be used to measure the distances between samples during classification.

Other future works could incorporate elements from other methods into the one presented in this work and vice-versa. For instance, a descriptor could be built with information from block matching trajectories, where the blocks tracked enclose certain feature points. Or, as previously stated, both trajectories and/or histograms of orientation could be used as features in a method based in bag-of-words.

Metric learning is also a topic that calls for further exploration, since we have made a very straightforward use of it in our method. We have indication that its use in controlled scenes could be very interesting, given other possibilities of adaptation. In this work, we suggested just one simple adaptation scheme. Moreover, we account that is necessary to more appropriately model the problem to more complex scenarios, since results for UCF11 have degraded considerably with the use of Metric Learning. Both the tensor representation of the descriptor and the metric matrix from Metric Learning are positive semi-definite matrices expressing transformations in Euclidean spaces, so we would suggest a more in-depth investigation of possibly other shared properties between the two, which could in turn lead to other more creative uses of Metric Learning in the method. Still, the results achieved, considering how small the descriptors become, are to be considered a positive outcome.

In summary, our remarks concerning the hypotheses of this work are:

- Trajectories based solely on block matching information are able to produce a motion descriptor with efficiency and efficacy for simple, controlled scenes. Furthermore, they are viable features to be incorporated into a more complex method.

- Histograms of relative displacements are able to describe motion in a frame sequence, as evidenced by our results being comparable to those of more complex methods that employ other features.

- Metric learning, despite not being able to improve the separability of the motion descriptors in the experiments we conducted, is still subject to further exploration. We have also found that Metric Learning greatly improves the compactness of descriptors without being severely taxing on recognition rates, for simple scenarios like the ones in the KTH dataset.

REFERENCES

CAETANO, F. A. **A Video Descriptor using Orientation Tensors and Shape-based Trajectory Clustering**. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, Juiz de Fora, Brazil, 2015.

CHAN, M.; YU, Y.; CONSTANTINIDES, A. Variable size block matching motion compensation with applications to video coding. In: **IEEE inproceedings 1990**, 1990. v. 137, No.4, p. 205–212.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, 1995. ISSN 1573-0565.

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

DALAL, N.; TRIGGS, B.; SCHMID, C. Human detection using oriented histograms of flow and appearance. In: **European Conference on Computer Vision (ECCV)**, 2006. p. 428–441. ISBN 978-3-540-33835-2.

DAVIS, J. V.; DHILLON, I. S. Structured metric learning for high dimensional problems. In: **International Conference on Knowledge Discovery and Data Mining (SIGKDD)**, 2008. p. 195–203. ISBN 978-1-60558-193-4.

DAVIS, J. V.; KULIS, B.; JAIN, P.; SRA, S.; DHILLON, I. S. Information-theoretic metric learning. In: **International Conference on Machine Learning (ICML)**, 2007. p. 209–216. ISBN 978-1-59593-793-3.

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

FERNANDO, B.; GAVVES, E.; ORAMAS, M. J.; GHODRATI, A.; TUYTELAARS, T. Modeling video evolution for action recognition. In: **Conference on Computer Vision and Pattern Recognition (CVPR)**, 2015. p. 5378–5387. ISSN 1063-6919.

FIGUEIREDO, A. M. D. O.; RENHE, M. C.; MOTA, V. F.; SILVA, R. L. de Souza da; VIEIRA, M. B. A video self-descriptor based on sparse trajectory clustering. In: **International Conference on Computational Science and Its Applications (ICCSA)**, 2016. p. 571–583.

FIGUEIREDO, A. M. de O. **A Video Self-descriptor based on Sparse Trajectory Clustering**. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, Juiz de Fora, Brazil, 2015.

GIBSON, J. J. **The Perception of the Visual World**. 1. ed., 1950.

GUILLAUMIN, M.; VERBEEK, J.; SCHMID, C. Is that you? metric learning approaches for face identification. In: **IEEE International Conference on Computer Vision (ICCV)**, 2009. p. 498–505. ISSN 1550-5499.

GUO, K.; ISHWAR, P.; KONRAD, J. Action recognition from video using feature covariance matrices. **IEEE Transactions on Image Processing**, v. 22, n. 6, p. 2479–2494, June 2013. ISSN 1057-7149.

HA, J.; ROSSBACH, C. J.; DAVIS, J. V.; ROY, I.; RAMADAN, H. E.; PORTER, D. E.; CHEN, D. L.; WITCHEL, E. Improved error reporting for software that uses black-box components. In: **Conference on Programming Language Design and Implementation**, 2007. p. 101–111. ISBN 978-1-59593-633-2.

HORN, B. K.; SCHUNCK, B. G. Determining optical flow. **Artificial Intelligence**, v. 17, n. 1, p. 185–203, 1981. ISSN 0004-3702.

HOROWITZ, S. L.; PAVLIDIS, T. Picture segmentation by a tree traversal algorithm. **J. ACM**, ACM, New York, NY, USA, v. 23, n. 2, p. 368–388, abr. 1976. ISSN 0004-5411.

ITSEEZ. **Open Source Computer Vision Library**. 2015. `https://github.com/itseez/opencv`.

JAIN, J. R.; JAIN, A. K. Displacement measurement and its application in interframe image coding. **IEEE Transactions on Communications**, COM-29, No 12, p. 1799–1808, 1981.

JAIN, M.; GEMERT, J. C. van; SNOEK, C. G. M. What do 15,000 object categories tell us about classifying and localizing actions? In: **Conference on Computer Vision and Pattern Recognition (CVPR)**, 2015. p. 46–55. ISSN 1063-6919.

JAIN, M.; JEGOU, H.; BOUTHEMY, P. Better exploiting motion for better action recognition. In: **Conference on Computer Vision and Pattern Recognition (CVPR)**, 2013. p. 2555–2562. ISSN 1063-6919.

KINDLMANN, G. Superquadric tensor glyphs. In: **Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym)**, 2004. p. 147–154.

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

KNUTSSON, H. Representing local structure using tensors. In: **Scandinavian Conference on Image Analysis (SCIA)**, 1989. p. 244–251.

KONIUSZ, P.; CHERIAN, A.; PORIKLI, F. Tensor representations via kernel linearization for action recognition from 3d skeletons. **arXiv preprint arXiv:1604.00239**, 2016.

LAW, M. T.; THOME, N.; CORD, M. Quadruplet-wise image similarity learning. In: **IEEE International Conference on Computer Vision (ICCV)**, 2013. p. 249–256. ISSN 1550-5499.

LI, R.; ZENG, B.; LIOU, M. L. A new three-step search algorithm for block motion estimation. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 4, n. 4, p. 438–442, 1994.

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

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

MOTA, V. F.; PEREZ, E. d. A.; MACIEL, L. M.; VIEIRA, M. B.; GOSSELIN, P.-H. A tensor motion descriptor based on histograms of gradients and optical flow. **Pattern Recognition Letters**, v. 31, p. 85–91, 2013.

MOTA, V. F.; PEREZ, E. d. A.; VIEIRA, M. B.; MACIEL, L. M.; PRECIOSO, F.; GOSSELIN, P. H. A tensor based on optical flow for global description of motion in videos. In: IEEE. **Conference on Graphics, Patterns and Images (SIBGRAPI)**, 2012. p. 298–301.

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

MURALIDHAR C.B. RAMA RAO, I. R. K. P. Efficient architecture for variable block size motion estimation of h.264 video encoder. **International Conference on Solid-State and Integrated Circuit (ICSIC)**, v. 32, p. 6, 2012.

OLIVEIRA, F. L. M. de; VIEIRA, M. B. Variable size block matching trajectories for human action recognition. In: **International Conference on Computational Science and Its Applications (ICCSA)**, 2015. p. 283–297. ISBN 978-3-319-21404-7.

PARIKH, D.; GRAUMAN, K. Relative attributes. In: **IEEE International Conference on Computer Vision (ICCV)**, 2011. p. 503–510. ISSN 1550-5499.

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

PO, L.-M.; MA, W.-C. A novel four-step search algorithm for fast block motion estimation. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 6, n. 3, p. 313–317, 1996.

POPPE, R. A survey on vision-based human action recognition. **Image and Vision Computing**, Butterworth-Heinemann, Newton, MA, USA, v. 28, n. 6, p. 976–990, 2010. ISSN 0262-8856.

PURI, A.; HANG, H.; SCHILLING, D. Interframe coding with variable block-size motion compensation. In: **IEEE Global Telecommunication Conference**, 1987. p. 65–69.

SANCHEZ, J.; PERRONNIN, F.; MENSINK, T.; VERBEEK, J. **Image Classification with the Fisher Vector: Theory and Practice**, 2013.

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

SCHULTZ, M.; JOACHIMS, T. Learning a distance metric from relative comparisons. In: **Advances in Neural Information Processing Systems (NIPS)**, 2004. p. 41–48.

SLANEY, M.; WEINBERGER, K.; WHITE, W. Learning a metric for music similarity. In: **International Conference on Music Information Retrieval (ISMIR)**, 2008. p. 313–318.

TRAN, D.; SOROKIN, A. Human activity recognition with metric learning. In: **European Conference on Computer Vision (ECCV)**, 2008. p. 548–561. ISBN 978-3-540-88682-2.

WANG, H.; KLASER, A.; SCHMID, C.; LIU, C.-L. Action recognition by dense trajectories. In: **Conference on Computer Vision and Pattern Recognition (CVPR)**, 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. **International Journal of Computer Vision**, v. 103, n. 1, p. 60–79, 2013.

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

WANG, L.; QIAO, Y.; TANG, X. Action recognition with trajectory-pooled deep-convolutional descriptors. In: **Conference on Computer Vision and Pattern Recognition (CVPR)**, 2015. p. 4305–4314.

WEINBERGER, K. Q.; SAUL, L. K. Distance metric learning for large margin nearest neighbor classification. **Journal of Machine Learning Research**, JMLR.org, v. 10, p. 207–244, jun. 2009. ISSN 1532-4435.

WESTIN, C.-F. **A Tensor Framework for Multidimensional Signal Processing**. Tese (Doutorado) — Linköping University, Linköping, Sweden, 1994.

XING, E. P.; NG, A. Y.; JORDAN, M. I.; RUSSELL, S. Distance metric learning, with application to clustering with side-information. In: **Advances in Neural Information Processing Systems (NIPS)**, 2003. p. 505–512.

YUAN, C.; HU, W.; LI, X.; MAYBANK, S.; LUO, G. Human action recognition under log-euclidean riemannian metric. In: ZHA, H.; TANIGUCHI, R.-i.; MAYBANK, S. (Ed.). **Asian Conference on Computer Vision (ACCV)**, 2010. p. 343–353. ISBN 978-3-642-12307-8.

ZELNIK-MANOR, L.; IRANI, M. Event-based analysis of video. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2001. v. 2, p. 123–130. ISSN 1063-6919.

ZHANG, Z.; WANG, C.; XIAO, B.; ZHOU, W.; LIU, S. Robust relative attributes for human action recognition. **Pattern Analysis and Applications**, v. 18, n. 1, p. 157–171, 2015. ISSN 1433-755X.

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