BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OPTICAL FLOW METHODS FOR VIDEO AND IMAGE SEGMENTATION

BACHELOR'S THESIS

AUTHOR	DIEGO TABERNERO SAN JOSÉ
SUPERVISOR	Ing. VÍTĚZSLAV BERAN, Ph.D.

BRNO 2019

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2018/2019

Bachelor's Thesis Specification

22103

Student:	**Tabernero Diego**
Programme:	Shortterm study BSc.
Title:	**Optical Flow Methods for Video and Image Segmentation**
Category:	Image Processing

Assignment:

1. Get acquainted with image and video processing methods focused on optical flow, feature extraction and statistics.
2. Design a system that segments the video sequences to stable parts and selected frames to background/foreground regions.
3. Prepare a suitable dataset and annotate these data.
4. Implement the system using the available libraries.
5. Evaluate the system on a prepared data set.
6. Document the methods, design, implementation, dataset and results in technical report.

Recommended literature:

- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, ISBN-13: 978-0495082521, 2007.
- Gary R. Bradski, Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*, ISBN 10: 0-596-51613-4, September 2008.

Requirements for the first semester:

- Items 1, 2, 3 and partially item 4.

Detailed formal requirements can be found at http://www.fit.vutbr.cz/info/szz/

Supervisor:	**Beran Vítězslav, Ing., Ph.D.**
Head of Department:	Černocký Jan, doc. Dr. Ing.
Beginning of work:	November 1, 2018
Submission deadline:	May 15, 2019
Approval date:	November 15, 2018

Abstract

This work presents a method to detect stable, or almost stable video-parts from the original video footage with large amount of fast large transformations (zooming, translation, rotation). It deals with approaches based on optical flow for getting information about the feature vectors of the video in each transition between frames. The classification methods used to determine if the consecutive frames are stable or not are machine learning methods like support vector machine (SVM) and K-means.

Keywords

Optical flow, machine learning, Lucas Kanade, Gunnar Fanerbäck, SVM, K-means

Reference

TABERNERO SAN JOSÉ, Diego. *Optical Flow Methods for Video and Image Segmentation.* Brno, 2019. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Vítězslav Beran, Ph.D.

Optical Flow Methods for Video and Image Segmentation

Declaration

I declare that I have prepared this Bachelor's thesis independently, under the supervision of Ing. Vítězslav Beran Ph.D. provided me with further information. I listed all of the literary sources and publications that I have used.

. . . .
Diego Tabernero San José
May 14, 2019

Acknowledgements

I would like to thank to my supervisor Vítězslav Beran to help and guide me in the development of this work.

Contents

Chapter 1

Introduction

Videos recorded from a handheld camera contain undesired features that affect directly to the quality of the output video. A slow motion in a video is not so important to the human eye but when is necessary to apply different video analysis, e.g. crowd analysis, to discover patterns inside the video, they do not work as well as a video with no harsh movement. Even when the problem is more severe with a high-frequency motion: jitter, translation, rotation or blurring in image sequences generate an annoying effect, hindering interpretation for both humans and computer vision systems seriously.

Video stabilization is the approach of negating the motion of the camera to produce a smooth motion across images of a shaky video stream. Their techniques can be classified in 3 main groups [11] (see Figure 1.1).

The first, hardware motion sensors (mechanical stabilization), based on vibration feedback or damper which helps as a support for the camera. The second is the optical image stabilization which uses a set of floating lens to fix the direction and path length of the light. Finally the last one is the video post processing methods called digital image stabilization which analyzes the image motion and obtains models to compensate and smooth the undesired, contained motion.

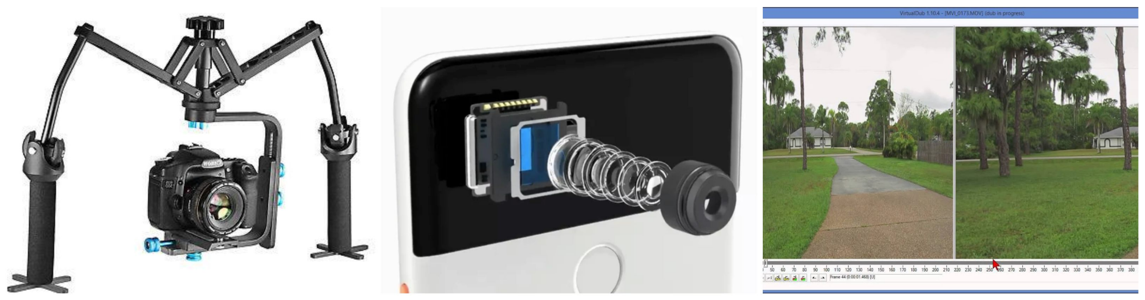


Figure 1.1: 3 approaches in video stabilization

The drawbacks about the mechanical stabilization and optical image stabilization are mainly their associated cost. In the second case, an own image stabilization system is required for each lens and can be only appropriated for some devices.

Digital stabilizers are a low-cost solution, not limited to any device that provide a good performance facing the problem via programmable software. It can also be easily maintained and improved though software updates, without removing any type of hardware.

In general there are two phases in the process of video stabilization: motion estimation to obtain the motion models and motion smoothing to counter the unwanted motion. On the other hand, stabilization video is not an easy task to solve and the practical results of different works have problems to carry it out, producing undesired results.

This work proposes a procedure to detect the video frames which do not contained a high-frequency motion using optical flow techniques to estimate the motion along the video frames, extracting and matching feature vectors between frames. This optical flow vectors will be used to create appropriate features through their magnitudes or modules, to proceed with these features as input of classification approaches employing supervised (support vector machine) and unsupervised (k-means) machine learning methods to determine the segments of video which do not suffer from unwanted motion or that could be fixed with a simple video stabilization post-processing approach.

Chapter 2

Theory

This section presents the theory as the base of the proposal of this work. It will explain in detail the different approaches that this work deals with and it will be exposed their theoretical fundaments for a better understanding of the work.

2.1 Crowd analysis

The set of problems of discovering the different, hidden patterns in datasets videos to recognize the changes in the behavior of a particular group of objects, especially people, are called crowd analysis.

The concept of crowd is common inside researches carry out in disciplines like psychology or sociology, but from time ago it has taken importance and has become a growing role in computer vision for surveillance systems. The increase of threats in social security, has made that video surveillance plays an important role.

The types of crowd could be different and is not the same to analyze a spread crowd with a low level of interaction than a high-dense crowd. In the last case, depending on the level of interaction and the organization of the crowd, the task could become complicated. A case of a crowd running a marathon (moving in the same direction) is easier if the goal is to detect any strange event like a person moving in a distinct direction when the rest of runners are running in the same direction, but if the task is to track a concrete person then the problem is complicated: a group of people share the features of the target (e.g the direction, and the more proximate runners almost the position).

In other kind of situations with a high-dense crowd like in a rally, the people are organised in front of the scenario, so here estimate the number of people who attended could be a easier task.

In general is more handable a sparse crowd than a dense crowd. It could be driven to undesired results, to model the crowd motion in environments with high interaction between the people, the different roles that the individuals can play in that situation or random objects with impact in the analysis. It becomes a difficult task and a challenge nowadays.

Depends on what pattern is interested in, the problem could be discomposed in some subproblems.

2.1.1 Estimate crowd density

The problem is based on estimating the number of people in a crowd attending to a particular event: a political rally, protest, sport event... Some of these situations show a difficulty in the task of estimating the crowd size, since having an entrance allow certain grade of control of the crowd and propose an easier solution to the problem.

Moreover, the result of applying these methods do not stop being an estimation, and therefore the offered number by different associations can vary greatly, creating controversy.

There are a group of methods that can be carry out the task: do a count of each person in a crowd with detection-based methods, dividing the surface of the place to estimate the density of people creating a density map (Figure 2.1) or using approaches based on convolutional neural network (CNN) as it is proposed in the paper [6].

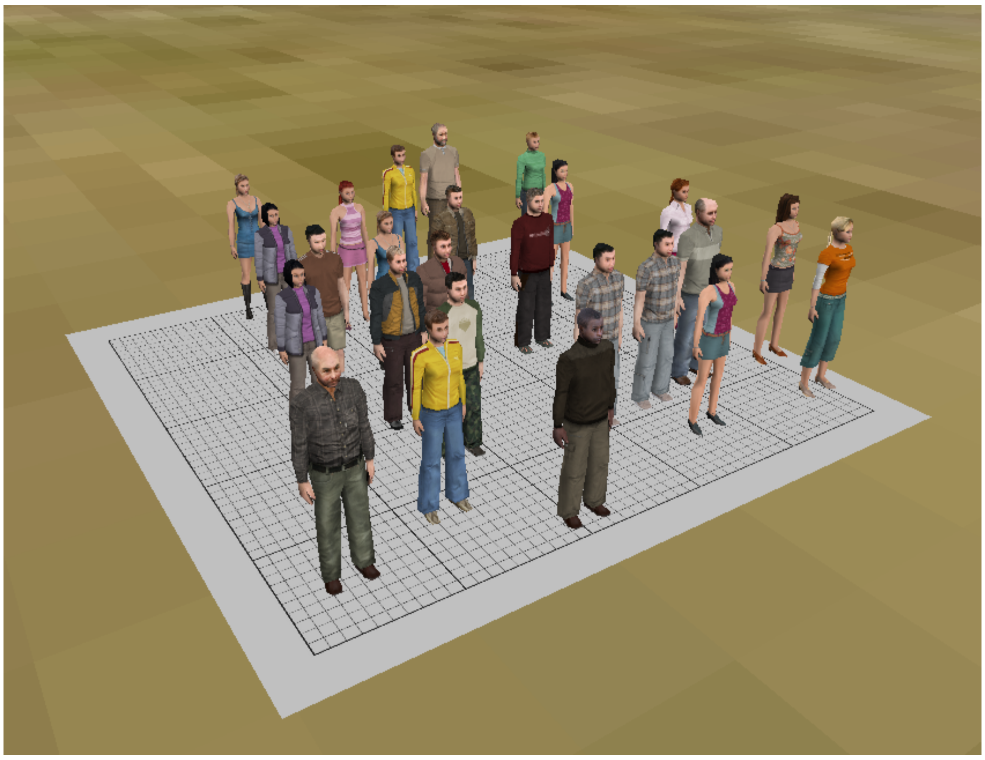


Figure 2.1: Crowd density estimation with grids[1]

2.1.2 Crowd segmentation

Crowd segmentation is the process of exploiting correlations of a sequence of video frames that contain similar people or objects. It is about detecting the boundaries to do a partition into multiple segments of pixels to simplify and analyse easier the sequence of frames.

This process does not give meaning at the partitions, and applying this method is possible to proceed to classify the segmentation using machine learning techniques to identify the different agents.

Some of the approaches found in the article [12] can be Region-Based Segmentation based on separate the objects with a value of a threshold, edge detection segmentation to

[1]Source: `http://www.gkstill.com/Support/crowd-density/CrowdDensity-1.html`

find the boundaries and edges to define the partitions or segmentation based on clustering which classifies the pixels into homogeneous clusters (Figure 2.2).

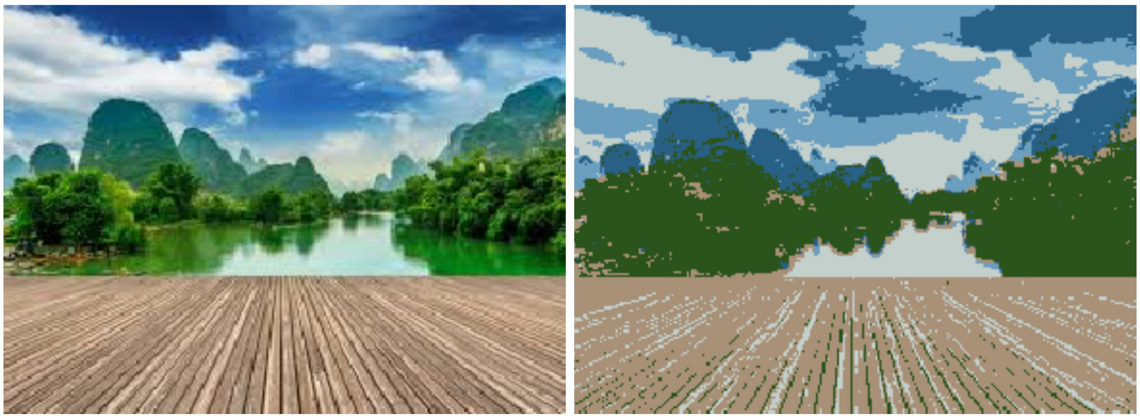


Figure 2.2: Segmentation based on clustering [12]

2.1.3 Track objects

Track objects is the procedure of locating in the space one or a group of objects, especially people, over time using a camera whose objective is to associate target people in consecutive video frames.

There are a set of methods to fulfill the purpose of tracking. In the paper [8] proposed an approach based on Minimum Mean Square Error (MMSE). It is robust in situations with change in brightness in complex crowd scenes.

On the other hand, there are approaches based on optical flow that are sensible to the change of grade of luminosity but offer a great results for tracking people. Some of them are going to be used in this paper, among are included Lucas Kanade (Figure 2.3) from the paper [7] and Gunnar Färneback from the paper [3].

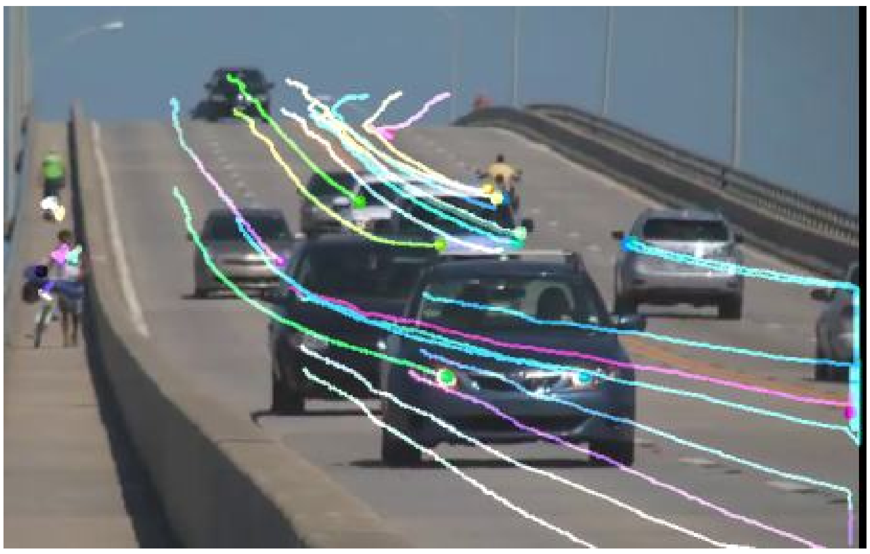


Figure 2.3: Cars tracking using optical flow by Lucas Kanade algorithm [2]

2.1.4 Video stabilization

Video stabilization is the process of improving the quality of the video negating the motion of the camera. They are a group of methods that can get reduce undesired inter-frame motion, compensating any angular, translation displacement or distortion contained in the video.

A general classification can be mechanical stabilization, optical stabilization and digital video stabilization as can be found in the paper [11]. The authors also present some methods to carry out the stabilization process: model or estimate the motion and, then smooth the motion (Figure 2.4).

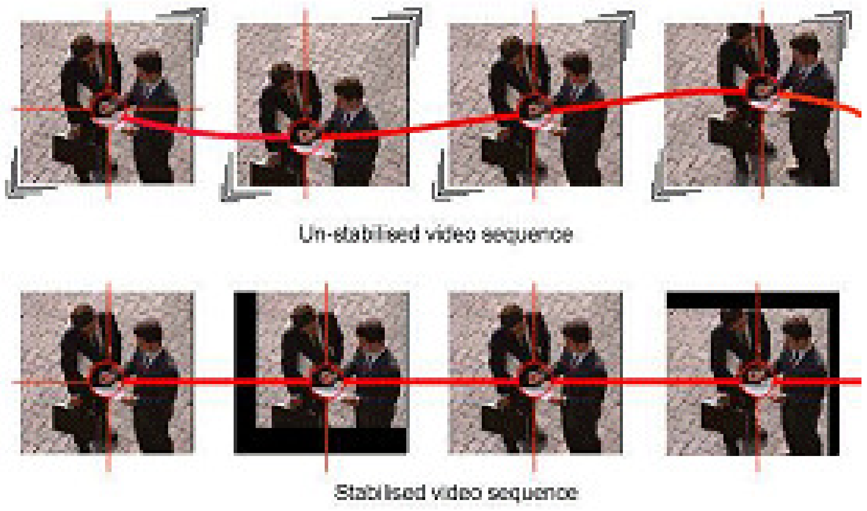


Figure 2.4: Smooth motion of video[3]

2.2 Optical flow

The estimation and modelling of the motion along the sequence of video frames is a challenge inside the area of computer vision. The real scene can be based on 3D vectors to represent the movement along the three axis but in a video is the projection onto the image plane getting a 2D vector. Moreover, to approximate the motion in a video, is necessary to work backwards, estimating these 2D vectors matching the pixels in the previous and the next frame.

These methods are called optical flow which can be defined as the identification of the patterns of apparent motion of image objects between two consecutive frames caused by the movement of object or camera. It is a 2D vector which represent the displacement, showing the movement of points from first frame to second, and son on (Figure 2.5). In the work [10] is presented some of the computation methods.

[2]Source: `https://docs.opencv.org/3.4/d7/d8b/tutorial_py_lucas_kanade.html`
[3]Source: `http://www.thedvshow.com/electronic-image-stabilization-and-picture-quality/`

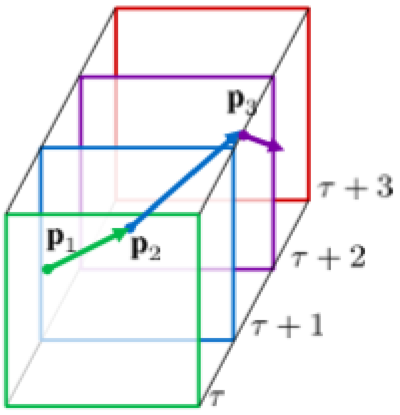


Figure 2.5: Optical flow movement[4]

Optical flow model has some assumptions. The most important is the pixel intensities of an object do not change between frames. In other words, the brightness in the image is constant. To fulfill this condition is necessary to consider the succession of frames the most proximate as possible with a short interval of time from t_0 to t_1, where the first and second time is from the previous and next frame, respectively.

It will be called the function I as the intensity or brightness of the image using the parameters x,y and t as the position at time t. Also, Δ is used to indicate the variation of the variable that follows it. The formula of constancy of brightness can be expressed as:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) \approx I(x, y, t) \tag{2.1}$$

Next step is applied a Taylor series expansion to the previous formula considering the small movement, obtaining:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{x}\Delta x + \frac{\partial I}{y}\Delta y + \frac{\partial I}{t}\Delta t + h.o.t \tag{2.2}$$

where $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the partial derivatives of the image parameters. The last part of the equation can be divided by ∂t and equal to 0 to get the following expression:

$$\frac{\partial I}{x}v_x + \frac{\partial I}{y}v_y + \frac{\partial I}{t} = 0 \tag{2.3}$$

that can be express in matrix form:

$$\nabla \mathbf{I} \bullet \mathbf{v} = -I_t \tag{2.4}$$

Thereby, the spacial (∇I) and temporal gradient (I_t), along with the optical flow vectors ($\mathbf{v} = (u, v)$) are achieved.

Although, they can not be computed since there are one equation and two unknowns. The problem is underdetermined and it will be necessary to add some constrain provided by different optical flow approaches and explained in the following sections.

[4]Source: http://mithril-ntu.github.io/L7_PAPER/

2.2.1 Sparse method

These approaches are designed to work on features or points of interest like edges or corners that can be found in the image. To this potential points are computed the optical flow vectors and can be found applying approaches as Harris corner detector [4] or Shi–Tomasi corner detection [13] a improvement of the Harris corner detector. These algorithms give more priority to the processing time in comparison with dense methods. This fact makes the sparse method a suitable method to work on real-time applications.

Inside this group is the popular method of Lucas-Kanade [7]. This procedure proposed a way of solving the optical flow equations based on the local differential technique. It proposes solving the optical flow vector with the assumption of considering that the vector has approximately a similar motion in neighbouring pixels. Thus, it adds this constraint to solve the optical flow equations for the pixels belong to this neighbourhood using the least squares criterion.

$$\nabla \mathbf{I}(p_j) \bullet \mathbf{v} = -I_t(p_j), \forall j = 1, ..., n \tag{2.5}$$

The selected set of neighbours pixels $\mathbf{p}$ contain n points. Each one will be assigned a weight that will represent the distance to the pixel under consideration. Thus, a farther neighbour pixel will be associated a smaller weight with the purpose of reducing its importance.

The matrix of weights is a diagonal matrix, such that:

$$W = diag[W(p_j)], \forall j = 1, ...n \tag{2.6}$$

In a matrix form, the problem of least square can be solved in the following way:

$$A^T W^2 A v = A^T W^2 b \tag{2.7}$$

which calculating the inverse matrix of $A^T W^2 A$, the optical flow vectors can be computed as:

$$v = (A^T W^2 A)^{-1} A^T W^2 b \tag{2.8}$$

where A is the matrix of gradient of the parameters x and y of all considering neighbours, W the matrix of weights and b the gradient of the parameter time.

In summary the method computes the optical flow vectors as follows:

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \sum W^2 I_x(\mathbf{p})^2 & \sum W^2 I_x(\mathbf{p}) I_y(\mathbf{p}) \\ \sum W^2 I_x(\mathbf{p}) I_y(\mathbf{p}) & \sum W^2 I_y(\mathbf{p})^2 \end{pmatrix} \begin{pmatrix} -\sum W^2 I_x(\mathbf{p}) I_t(\mathbf{p}) \\ -\sum W^2 I_y(\mathbf{p}) I_t(\mathbf{p}) \end{pmatrix} \tag{2.9}$$

2.2.2 Dense method

This group of optical flow approaches are focused on computing the optical flow vectors for each single pixel of the image. This task has a considerable cost of time, but in exchange it gets a larger information to understand the motion along the video frames.

Inside this group it can be found the method of Gunnar Farnebäck [3] to produce the optical flow vectors. The method makes use of the technique quadratic polynomial expansion to estimate a window or neighbourhood of pixels of two frames. Then, learning from how the polynomial transforms with the motion, it estimates the displacement parameters from the polynomial expansion coefficients.

To reach the goal of computing the displacement parameters, first it is used the equation of quadratic polynomial expansion to approximate each pixel neighbourhood:

$$f_1(x) = x^T A_1 x + b_1^T x + c_1 \tag{2.10}$$

with f_1 a signal, A a symmetric matrix, b a vector and c an scalar. It is constructed a new signal f_2 with a global displacement d:

$$\begin{aligned} f_2(x) &= f_1(x - d) \\ &= (x - d)^T A_1 (x - d) + b_1^T (x - d) + c_1 \\ &= x^T A_1 x + (b_1 - 2A_1 d)^T x + d^T A_1 d - b_1^T d + c_1 \\ &= x^T A_2 x + b_2^T x + c_2 \end{aligned} \tag{2.11}$$

From the equation 2.11 is possible to observe the following characteristics, with special attention in the second one:

- $A_2 = A_1$
- $b_2 = b_1 - 2A_1 d$
- $c_2 = d^T A_1 d - b_1^T d + c_1$

If the matrix A_1 is non-singular, given a translation d is possible to compute the displacement, i.e. the optical flow, as follows:

$$d = -\frac{1}{2} A_1^{-1} (b_2 - b_1) \tag{2.12}$$

2.3 Machine learning

Machine learning belongs to the field of artificial intelligence and is the set of techniques that answer to the question of how to automatically is possible to improve the performance of computer programs in a determined task through the experience [9].

It involves different steps to carry out its goal. One of them is the search of the optimal hypothesis that best fit to the observed data and its prior knowledge, from which it extracts the experience. This hypothesis is chosen from the whole set of possible hypothesis contained in the hypothesis space, which usually its size determine the computational complexity.

The representation of these hypothesis can be doing by decision trees, artificial neural networks, decision rules, linear functions, instance-based, and a few others. For learning different kinds of target functions, these different hypothesis representations are appropriate.

The base of machine learning methods come from different fields, that is why it is possible to call it multidisciplinary. Among them, it can be found statistics and mathematics, artificial intelligence, philosophy, information theory (tree decision), biology (artificial neural network) or computational complexity (determine the amount of necessary resources to run the algorithm).

These methods has a wide use inside many and very varied applications, ranging from fraud detection from the data of card transactions to the creation of customers profiles with the stored data about their preferences in their purchases in a determined website with their account.

This work deals with two kinds of machine learning approaches, that will be explained in the following two subsections. They are called supervised and unsupervised, and inside each one, there is a method which will be used in the classification step of the proposal. Concretely, support vector machine (supervised) and k-means (unsupervised).

2.3.1 Supervised methods

Classification learning is called supervised since the scheme operates under supervision by being provided with the actual outcome for each of the training examples. This outcome is called the class or label of the example.

In this approach, the algorithm generates a function to do a correspondence between the input and the desired output (class).

The success of classification learning can be calculated by testing the learned concept description on an independent set of test data for which the true classifications are known but not made available to the machine. The success rate on test data gives an objective measure of how well the concept has been learned.

Supervised methods can be divided in classification or regression depend on if the class is restricted to a number of categories or, unlike, it can be a number from a range [15]. Figure 2.6 shows the graphical difference, while in the case of classification the line separates both groups, for a regression the line is adjusted to the points.

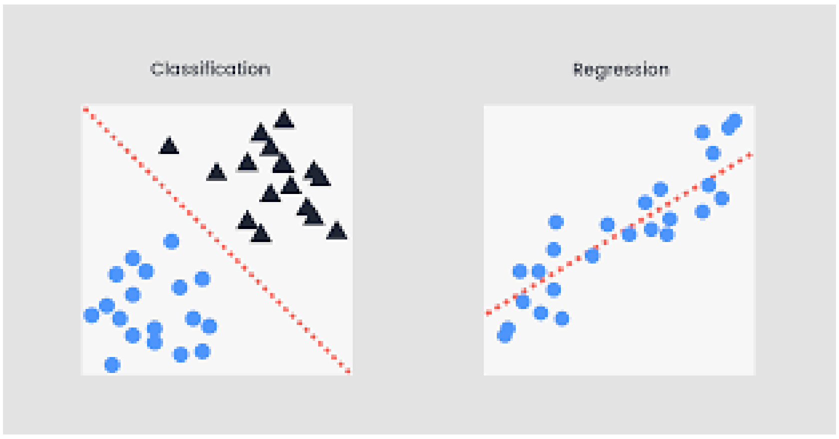


Figure 2.6: Classification vs. Regression[5]

Inside this group is the support vector machine method. It is a supervised and classification method which goal is to build a hyperplane of dimension N that separate the examples as widely as possible [5, ch. 12].

It selects a small number of instances from each class to become which is the so-called support vectors as can be seen in Figure 2.7. They will be the critical boundaries of the classificator located at exactly a of distance M/2 from the hyperplane, where M is the margin or distance between the support vectors of each class.

[5]Source: https://www.seebo.com/machine-learning-ai-manufacturing/

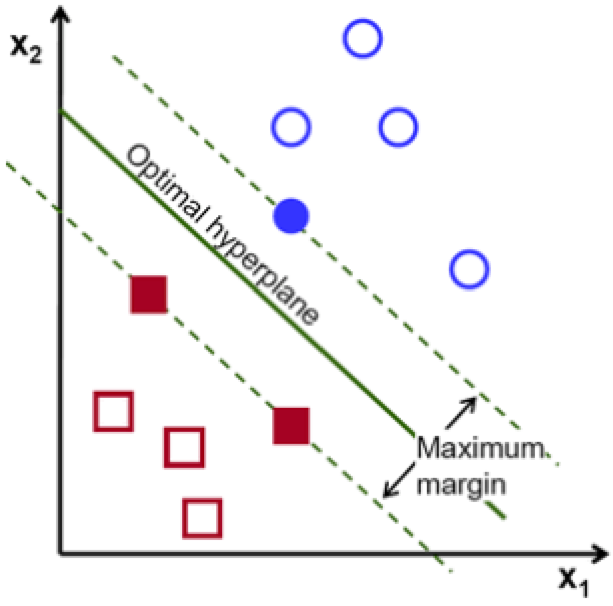


Figure 2.7: Support vectors at distance M/2 from the hyperplane[6]

For a set of n examples $x_i \in \mathbb{R}^p$ where there is 2 classes $y_i \in \{-1, 1\}$ and exist an hiperplane $\{x : \beta_0 + \beta^T x_i > 0\}$ that can be separate them linearly as:

$$(\beta_0 + \beta^T x_i) y_i > 0, \forall i = 1, ..., n \tag{2.13}$$

The distance of the point x to $\{x : \beta_0 + \beta^T x = 0\}$ is $\pm\frac{1}{||\beta||}(\beta_0 + \beta^T x) = 0$. If is taken $||\beta|| = 1$ and data is linearly separable, then the distance from the point x_i to the hyperplane will be $y_i(\beta_0 + \beta^T x_i)$, always positive as can be shown in 2.13.

The problem can be posed to find the hyperplane in the sense of larger margin M as:

$$max\ M\ /\ y_i(\beta_0 + \beta^T x_i) \geq M,\ \beta_0, \beta\ with\ ||\beta|| = 1 \tag{2.14}$$

If is taken $||\beta|| = \frac{1}{M}$, then equation 2.14 could be simplify to $y_i(\beta_0 + \beta^T x_i) \geq 1$ and the distance of the examples in the margin will be $M = \frac{1}{||\beta||}$, verifying the support vectors $y_i(\beta_0 + \beta^T x_i) = 1$. They are the only examples which contribute in the computation of the margin.

2.3.2 Unsupervised methods

In this section is presented the opposite approach of supervised methods, called unsupervised since all process of modelling is carrying out throughout a set of examples formed by just the input without the label of desired result.

It does not have information about the categories of the examples, and thus, the algorithm is based on classify and separate the examples in groups, characterized by some pattern that the system found in their attributes. They usually use measures about the similarities or differences among the examples to determine the different clusters.

[6]Source: https://docs.opencv.org/3.4/d1/d73/tutorial_introduction_to_svm.html

The variety of unsupervised methods can belong to two main groups [2]:

- **Hierarchical**: It is built a hierarchy of clusters and the membership to a cluster in a determined level is conditioned by the membership to superior levels. They have two types of construction following the process shown in Figure 2.8:
  - **Agglomerative or bottom-up**: The construction is carried out from up to down. It starts out from cluster with just one example, and iteratively they are joint.
  - **Divisive or up-bottom**: The construction is carried out from down to up. At the beginning all the examples are in one single cluster and progressively they are splitted, descending in the hierarchy.

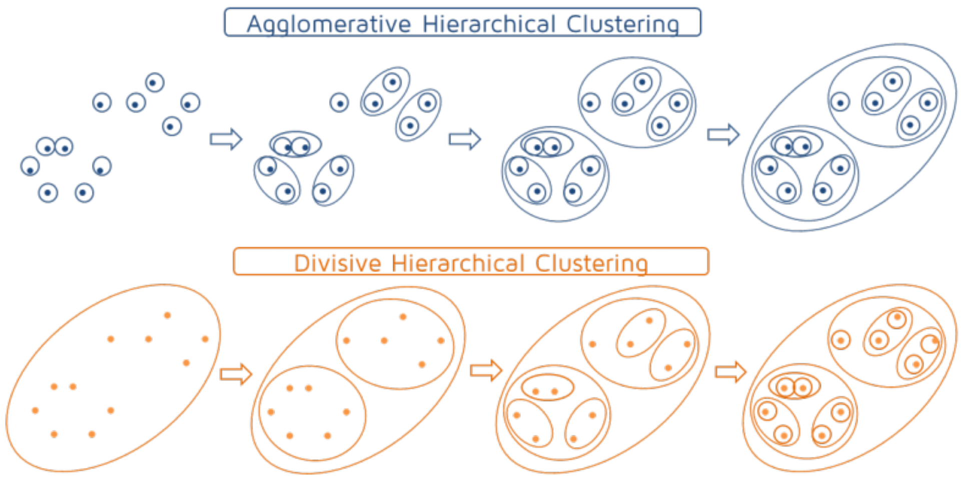

Agglomerative Hierarchical Clustering
Divisive Hierarchical Clustering

Figure 2.8: Hierarchical: Agglomerative vs. Divisive[7]

  -

- **Non-hierarchical**: They obtain a single partition of clusters through the optimization of some function.

The method used in this work is the k-means [5, ch. 14] which is unsupervised and non-hierarchical. The number k of clusters is fixed from the beginning of the algorithm and the problem can be posed as the pursuit of k optimal centers $m_1^*, m_2^*, ..., m_k^*$ to classify a set of n examples $x_1, x_2, ..., x_n \in \mathbb{R}^p$.

$$\{m_1^*, m_2^*, ..., m_k^*\} = arg \min_{m_1, m_2, ..., m_k} \sum_{i=1}^{n} ||x_i - m_j||^2 \tag{2.15}$$

This will create k regions corresponding to the different clusters which are obtained by proximity criteria to the k-means:

$$Cluster\ J = x_i \ \ ||x_i - m_J^*||^2 \leq ||x_i - m_j^*||^2, \ j \neq J \tag{2.16}$$

[7]Source: https://quantdare.com/hierarchical-clustering/

Chapter 3

Proposal

The aim of this chapter is to define the methodology to solve the problem of detection and classification of frames with no harsh movement belong to a video which presents different types of phenomenons of movement.

The proposed solution is composed of two parts. First, collect proper and enough features from the video frames. Second, this feature set will be used as input of classification methods.

The extraction of the features will be done by two different optical flow approaches which are explained in Section 3.2. They are called Lucas Kanade and Gunnar Farnebäck respectively, and their goal is to compute optical flow vectors between two frames.

After the video is processed with optical flow algorithms, the output will be the features that, depends on the classification method used, it is proposed to build different types features. In this work, there are 3 methods to classify the frames in stable or non-stabble: a classificator based on a threshold, support vector machine and k-means. These classification approaches are presented in Section 3.3 and the possible datasets used for these methods are presented in Section 3.1.

3.1 Data

This section is focused on explaining the characteristics of the handled data which will be used to process the input video dataset into a feature set. This computation will allow to obtain some measures along with the algorithms, making the evaluation and the testing of the considered methods to decide which of them work better.

At the beginning, non-stationary videos with some type of movement are available. They might contain a sequence of frames with some translations (Figure 3.1), rotations, blurrings or zoomings. The rest sequence of frames have a reasonable stationary state with no significant movement. A video is considered as set of frames sorted temporally, so the base are images which are stored though pixels with 3 color channels (RGB).

The videos are recorded from a high place to obtain a general view because in this way the optical flow algorithms can recognise better the movement along the video. In this case, the camera is in a helicopter recording the pedestrians walking in the surroundings of a football stadium (Figure 3.1).

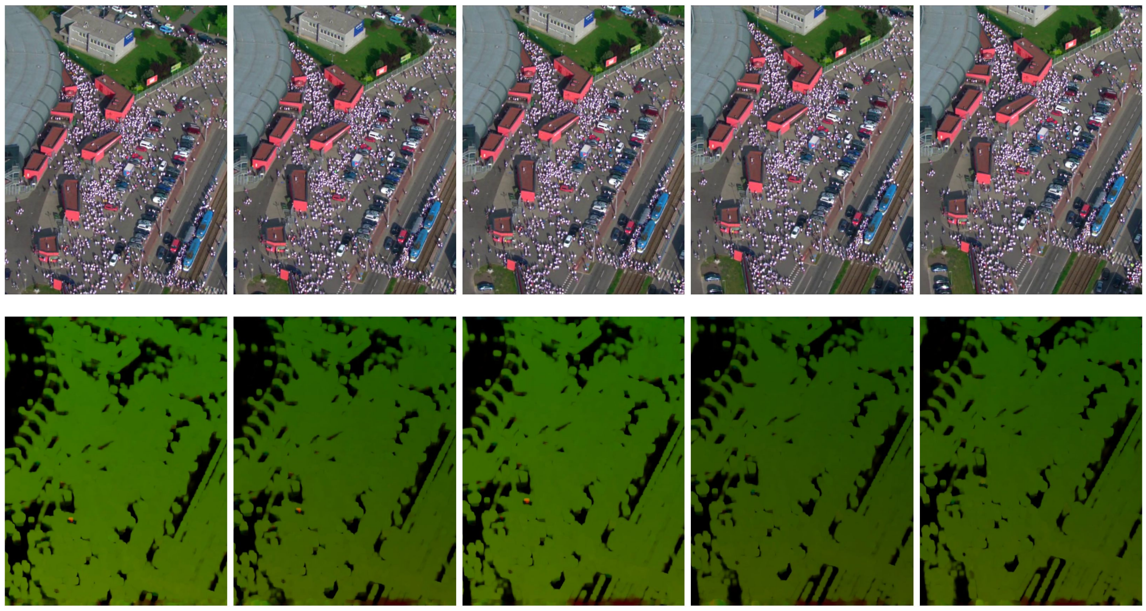

[]
[]

Figure 3.1: Sequence of 5 frames with movement of translation

To carry out the feature set is necessary to extract the magnitude or length of the optical flow vectors, this is, the displacement of each pixel between two consecutive frames. For that, is necessary to apply optical flow approaches to generate the vectors.

There are two proposed approaches: Farnebäck's algorithm, to get a vector for each pixel in the image (dense method) or Lucas Kanade algorithm selecting interesting points with Harris corner detector [4] or Shi-Tomasi corner detector [13], to get its vector with less time processing but with less number of vectors per image.

In both cases, the estimated vector contains the magnitude and direction given in cartesian coordinates. In this case is easy to calculate the magnitude applying the euclidean norm to get the module of the vector.

In a first proposed festure set, to register the quantity of movement inside the transition between two consecutive frames, it is defined the amount of movement as the sum of the modules of the optical flow vectors since the distances among two matching points in two consecutive frames are expected to be similar for background pixels, regardless of the camera motion direction. Thus, the sum of the modules of the vector is supposed to be small for the pair of frames with a reasonable stability in comparison with the pair of frames which presented clear evidences of movement.

These features will be used in a classification method based on estimating a threshold to get the highest success rate splitting the frames in two groups throughout the variable amount of motion.

In a second feature set that will be used with the machine learning methods of K-means and support vector machine, is necessary to develop a structure inside the data which contains more than a single variable. The applied procedure in this work is about getting the optical flow vectors and once the module of each vector is computed, it must count the number of times that the length of the vector is between 0 and 5, 5 and 10, 10 and 15, and so on, until the magnitude is 50 or more. In this way the values of the magnitude are divided in 10 bins and it is got a discretization of the variable to create features with 10 attributes that can be used in more sophisticated machine learning algorithms. Figure 3.2

shows the computed optical flow with a dense approach to get the histogram of magnitudes.

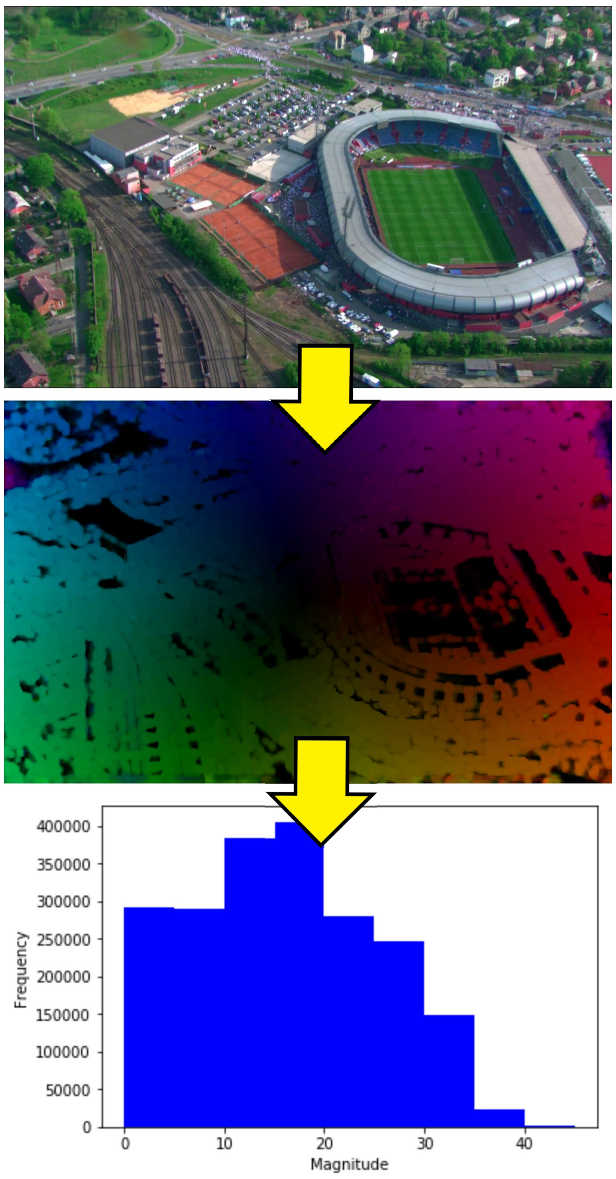


Figure 3.2: Dense optical flow and histogram of magnitudes

These features must contain a variable that indicates if the observation is stable or non-stable. It is called the class variable and it allows to calculate the success rate comparing the predictions of the different models with its real class. Moreover, it will be used when the model will be trained.

3.2 Extraction of features from optical flow vectors

This section proposes the use of two optical flow techniques to collect the information about the motion of a video. They are the sparse method of Lucas Kanade and the dense method of Gunnar Farnebäck to track some interesting points or all of them, respectively.

Both methods have their advantages and disadvantages. Lucas Kanade is focused in the time processing, selecting previously better, potential points to track, but less information is collected. On the other hand, Gunnar Farnebäck uses the whole set of points to track collecting the larger amount of information available (Figure 3.3), but the exactitude on the tracking process is not so accurate in all points.

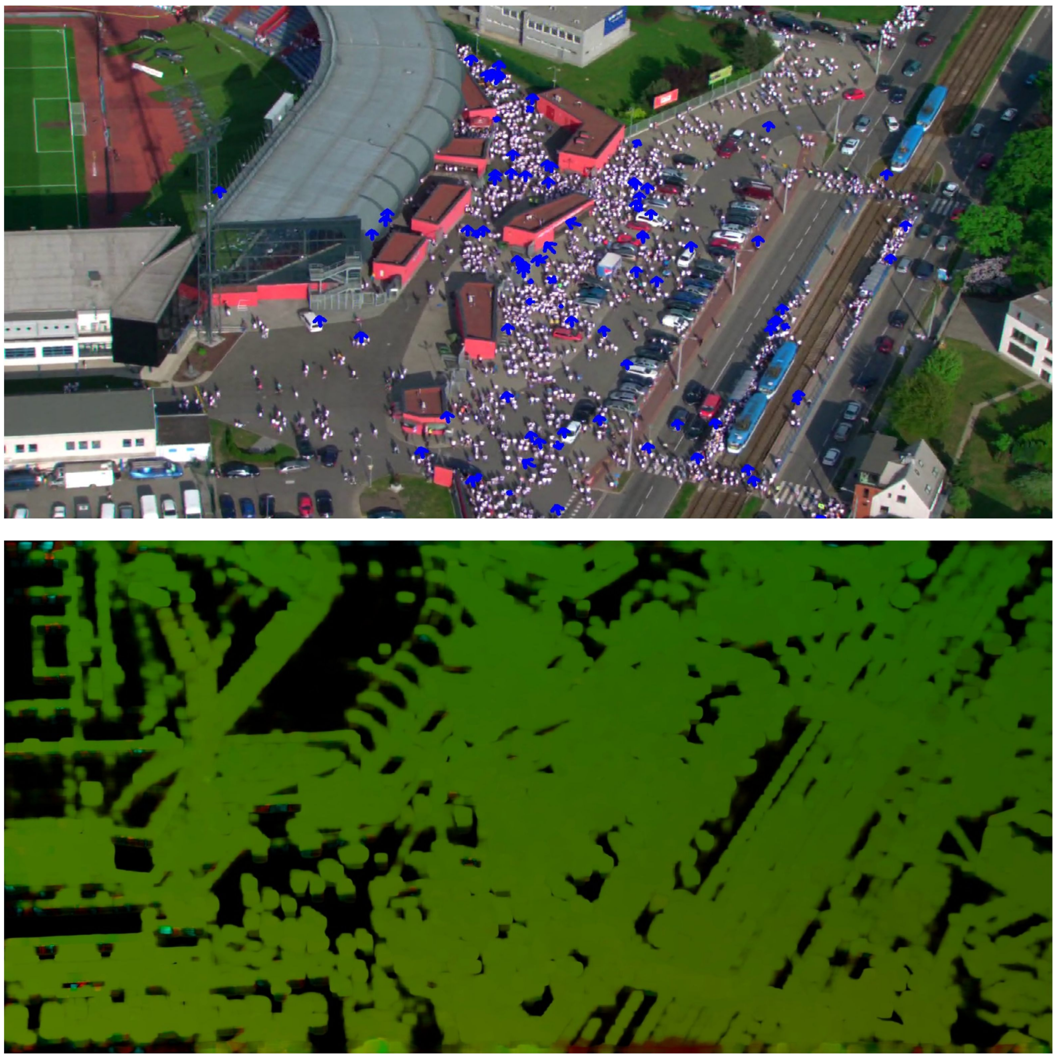


Figure 3.3: Optical flow vectors with sparse (up) and dense method (down)

After, these methods will be tested to see the variation on the success rate of the different classification models and determined which of them produce better features for the purpose of this work. The following sections introduce the methodology to apply these algorithms for computing as the amount of motion as for the histogram of magnitudes.

3.2.1 Amount of motion

The amount of motion is the considered feature to use as input in the problem of classification with threshold. It is the sum of the modules of the optical flow vectors and a representation of the quantity of motion that the transition among two frames includes.

Both optical flow methods are used to compute the amount of motion. They will be different estimations of the variable to see the impact in the success rate after testing the classification methods with each feature.

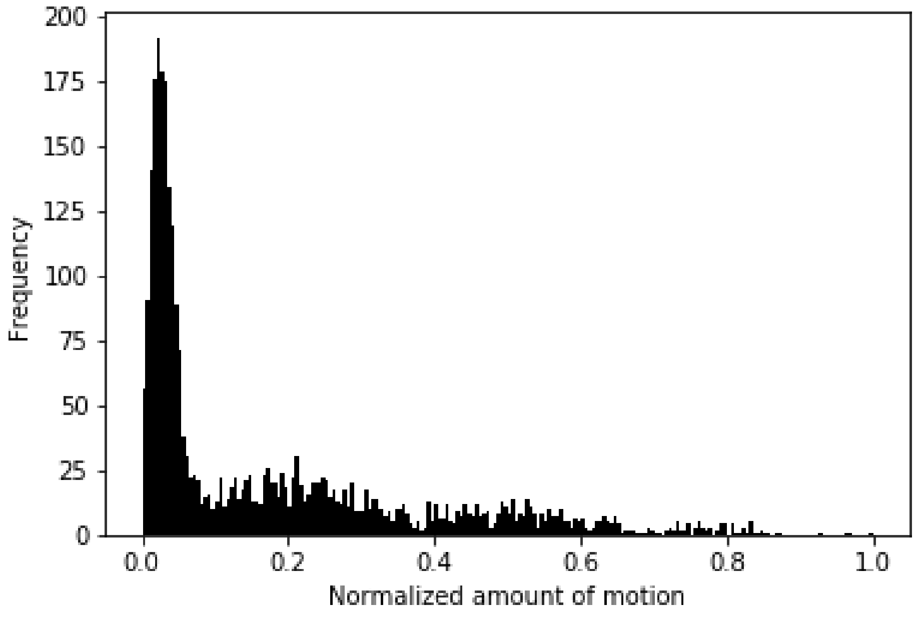


Figure 3.4: Normalized dense histogram of amount of motion in a sequence of 3000 video frames

3.2.2 Histogram of magnitudes

The histogram of the magnitudes is the considered feature to use as input with support vector machine and k-means approaches.

It is another way of registering the motion between two frames discretizing the possible values of the module of optical flow vectors, generating a p-dimensional feature. In this case, it is formed 10 attributes to work better with these approaches.

The histogram of magnitudes is achieved throughout the method of Lucas Kanade and Gunnar Farnebäck to compute the optical flow vectors.

In the first case, it has to deal with the issue of selecting potential points to track. It will be done in each frame, instead of letting the same set of points along the frames, because the goal is not tracking the points among the maximum number of frames if not getting the information between the transition of each two frames.

In the second case, the algorithm does not have this problem because it tracks the whole set of points of each frame and it returns the displacement of each pixel among the two frames. A example of the process for this method can be seen in Figure 3.2.

3.3 Classification

This section is about the classification procedure where from the extracted festures with optical flow approaches is carried out the training of the model to get one with the highest success rate for each method. From here is possible to do predictions using the model to classify unknown frames automatically.

In the assessment of the model it is used the technique cross validation to guarantee the independence between the training set and the test set [14]. It is based on the repetition about the calculation of the success rate over different partitions of the data.

In this case it is used the 10-fold cross validation where the data are splitted in training set, the 90% of the observations, and test set the 10%, respectively. It is repeated 10 times selecting the sets in this way, always in each iteration using a different test set as shown Figure 3.5.

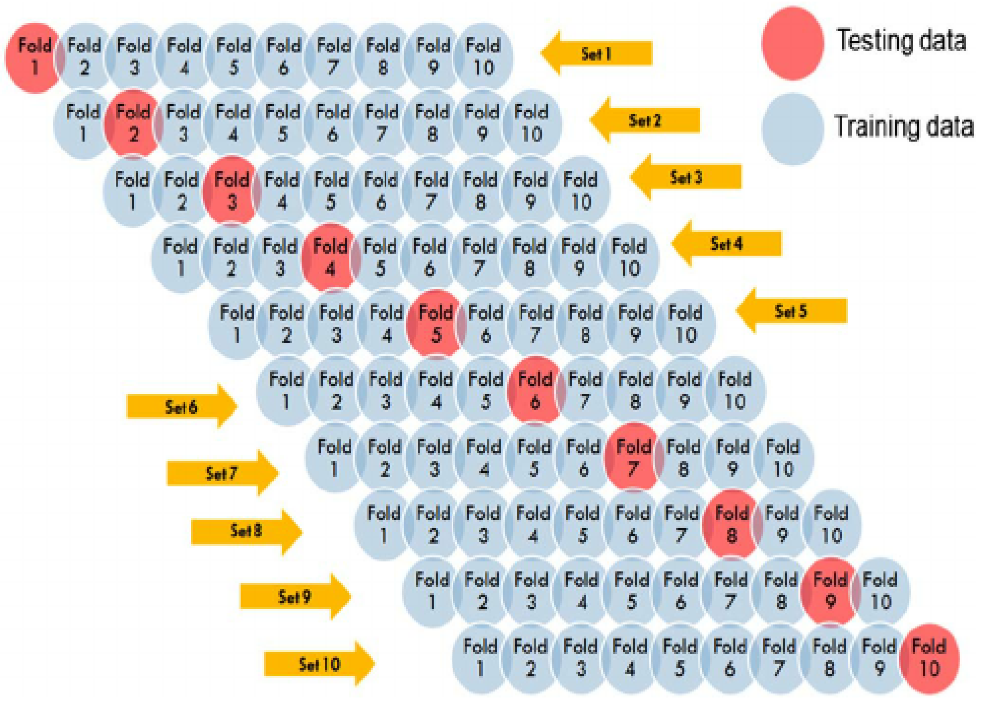


Figure 3.5: 10-fold cross validation[1]

With cross validation, it is achieved less bias in the estimation, with the reduction of the variance in the estimator of success rate.

3.3.1 Threshold

This method proposes the use of one variable to classify the video frames into stable or non-stable. This is called amount of motion and is the sum of the modules of all vectors contain in a frame.

[1]Source: https://www.researchgate.net/figure/10-fold-cross-validation-procedure_fig3_239386696

As a exploratory way at the beginning, it is suggested to draw the histogram of amount of motion and differentiate the frames by the group they belong with the label variable as it is shown in Figure 3.6.

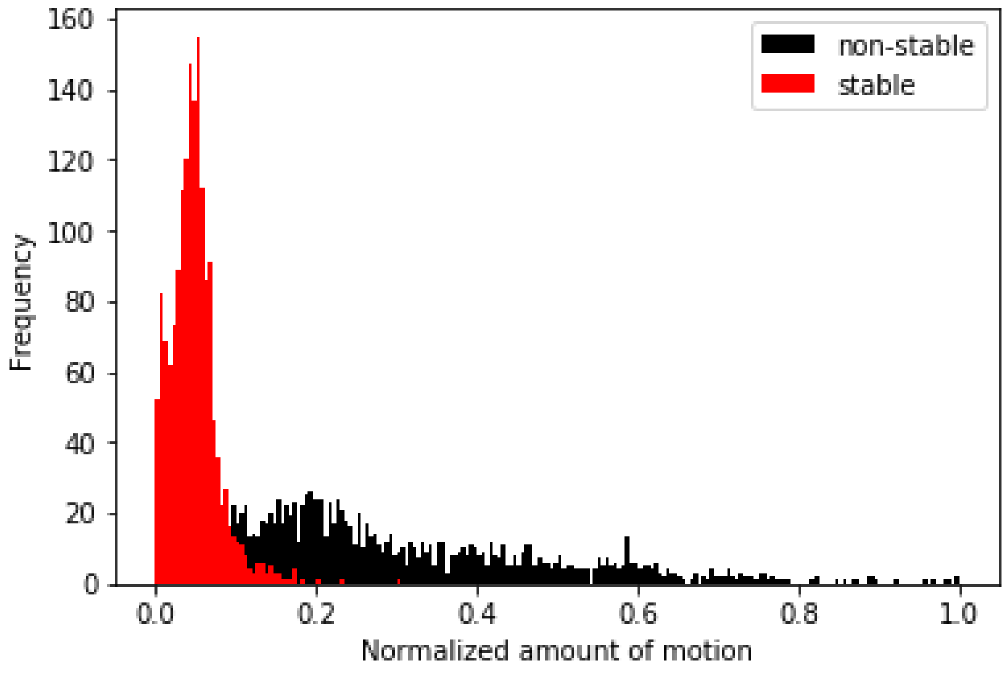


Figure 3.6: Normalized sparse histogram of amount of motion by type of frame

In this graphic is possible to see the stable frames in red color and the non-stable frames in black color. The most stable frames are situated between 0 and 0.15 while the non-stable are distributed along the histogram. Thus, in this image the optimum threshold must be between the values 0 and 0.15, and here is where it must be searched.

First, it is important to stratify the train set and the test set to get all samples have the same distribution than the population in the 10 iterations of the cross validation algorithm. In this case the train and test set must have 45.97% of stable frames and 54.03% of non-stable frames in each iteration. This is done to get a less standard deviation, and thus, an improvement in the accuracy of the estimation of the success rate.

This method explores the values of the threshold between 0 and 0.15 each 0.01, so in total 15 values of the threshold are checked in each of the ten iterations of the cross validation. For each value of the threshold, the model is trained through the training set: their observations are classified with the current threshold value as stable frames if the amount of motion of the frame is lower or equal than the threshold or non-stable frames if it is greater.

Then, with the classification of the training set, the predictions are compared with the actual group that the observations belong. It is used the label variable, to do this testing and obtain how many of the frames inside the training set were predicted correctly. A success rate is calculated with this number divided by the total size of training set and it will be used to determine if this threshold is the optimum for the selected training set.

This procedure is repeated with the different values of the threshold, and the final one will be which has the biggest value in the success rate in the training step. The selected model will be used to make the predictions with the test set and to do the assessment of this model in the same way that with the training set explained previously.

After 10 iterations of the cross validation it will be had 10 success rates and 10 thresholds. The final threshold and success rate will be the average of these 10 results.

About Figure 3.7, this is the output of the algorithm: a threshold (in blue) to separate both groups and make the best classification based on the amount of motion of the frames.

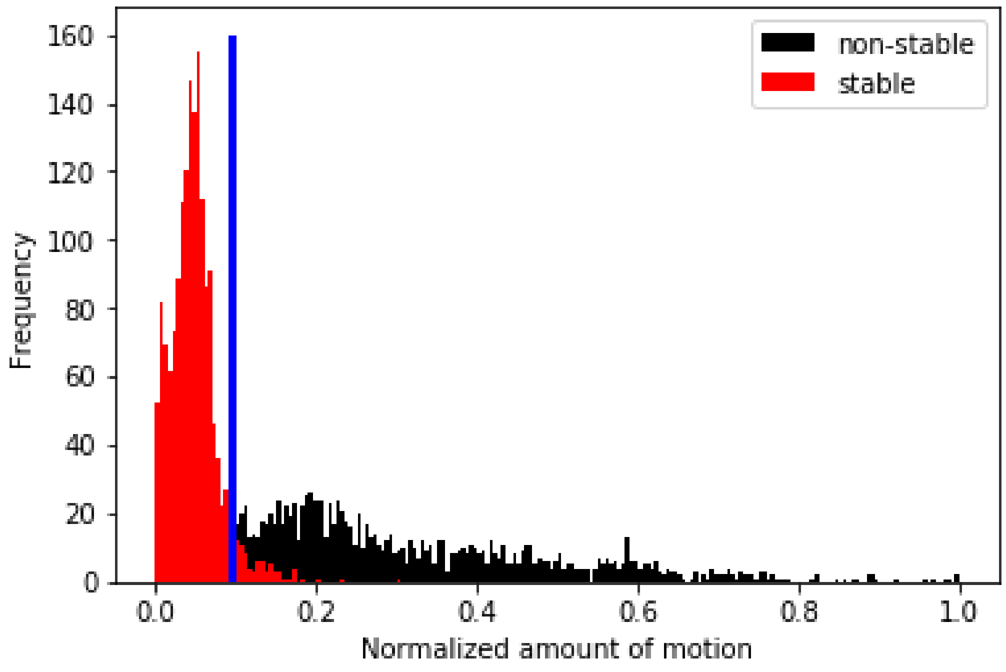


Figure 3.7: Normalized sparse histogram of amount of motion with the computed threshold

3.3.2 Support Vector Machine

Another approach to get a proper way of carrying out the classification of the frames is using the technique support vector machine. The basic idea of this method is about finding a hyperplane which provides the maximum separation between two separable groups in a linear way. The created partition in the space by the hyperplane is in the sense of maximum margin which means the maximum distance of the support points (observation what are exactly in the margin) to the hyperplane.

The extracted features for this method is composed of 10 attributes based on the magnitude of the optical flow vectors. When the optical flow vectors are estimated it is counted how many of them have a magnitude between 0 and 5, 5 and 10, and so on until the tenth attribute which is a lenght more than 50. In this way, it is had a dataset with enough variables (high dimensionality) to use as input for the support vector machine and thereby, have more information to discriminate better the observations.

With the histogram of magnitudes of each frame, the next step is to normalize the features to avoid that some variables have more importance than others when the optimal hyperplane is computed. It is proposed two types, both by columns.

In a first approach, the normalization is carried out taking the values of each variable to the range 0-1. It is a scaling of the features using the following formula:

$$X_{normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

Secondly, the standarization of the variables is about subtract the mean of the variable to the value and then divided by the standard deviation of the variable. This process gets standard scores which can be compared:

$$X_{standarized} = \frac{X - \mu_x}{\sqrt{\sigma_x^2}} \tag{3.2}$$

From this point, it is started the process of training the model to get the optimal hyperplane. To evaluate the model is used the 10-fold cross validation. In each iteration, as for the training set as for the test set, it is important to consider stratified samples to have in both sets the same distribution than the population. Other approach instead of stratification is a randomization, shuffling all observations firstly to avoid the extreme situation that they could be ordered.

After the model is computed, the test set is used to get a success rate for each iteration, and the final success rate of the model will be the average of the 10 success rate.

It is possible to try with approaches that supposed observations non-linearly separable. These are the so-called kernels doing a transformation of the initial attributes to a space of characteristics of greater dimension where the groups can be separated linearly [1]. The most popular kernels are the polynomial kernel as it is shown in Figure 3.8, perceptron kernel or radial basis kernel.

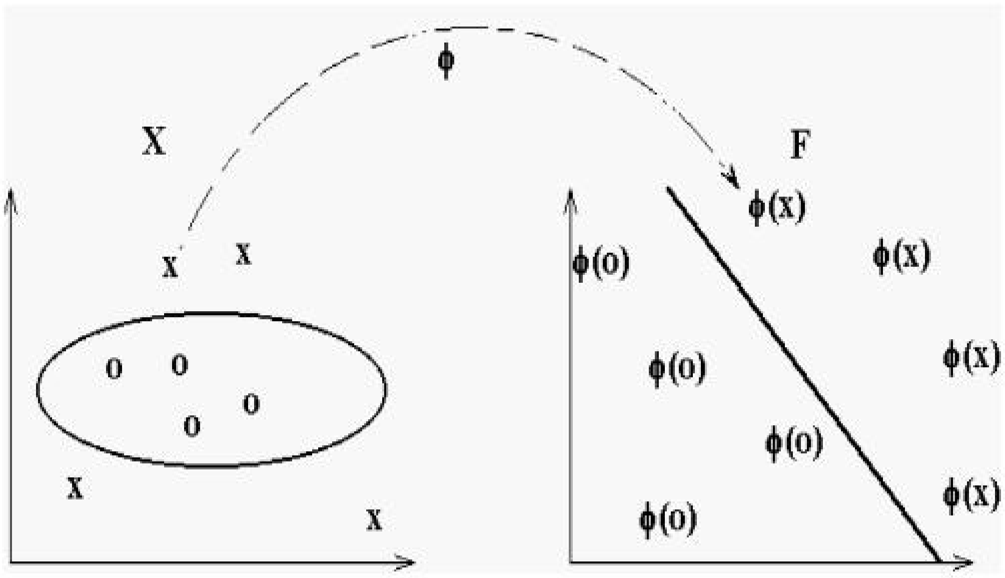


Figure 3.8: Attributes taken to the space of characteristics[2]

[2]Source: https://www.researchgate.net/figure/Kernel-trick_fig4_275346709

3.3.3 K-means

In this section is presented an unsupervised method to discover the possible groups that the frames can be classified. The K-means, as an unsupervised method, does not need the class variable to be trained but it will be used it to do the assessment of the algorithm.

The goal using K-means is discovering the cluster which contains the largest number of stable frames. It will be used different number of clusters, not only 2, to see the development of the success rate and see other subgroups that can be contain in the non-stable group: frames with translation, zooming, blurring, rotation, and different phenomenons that can happen in a non-stable frame. About the stable group, it is assume that it does not contain any subgroup.

It is proposed the use of 5, 10 and 25 groups where the observations will be classified. After the creation of the model, it will be explored the clusters, looking for the one with the largest number of stable frames since one assumption is that the stable group does not contain any subgroup and only the non-stable contains the subgroups. Therefore, when it is found this stable cluster, the rest will be considered as non-stable.

The k-means starts considering k random centers in the space and its goal is to find the k optimum centers, assigning each observation to the group with proximity criteria. In each iteration the centers are updated as the average of all the observations that belong in that moment. The algorithm will finish using stability criteria, this means, when there is no significant difference in the position of the centers between the previous iteration and the current one. Figure 3.9 shows the described process.

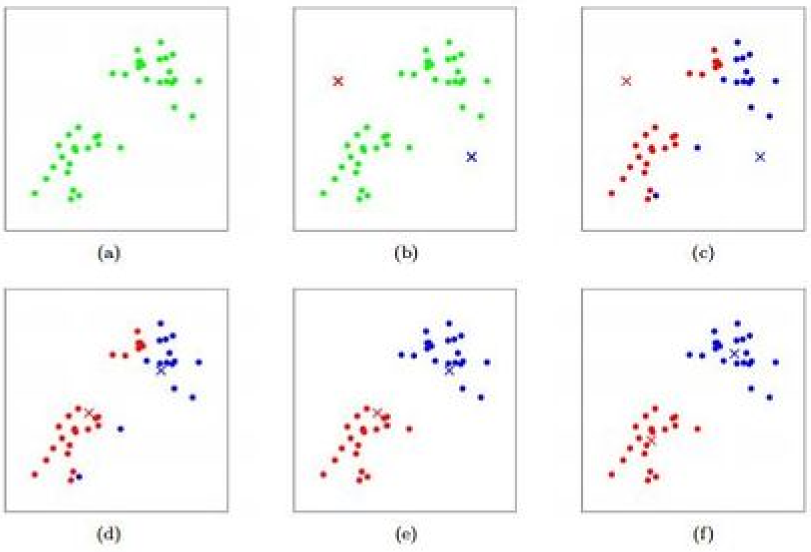


Figure 3.9: K-means procedure in 6 iterations[3]

Before the use of the algorithm is necessary to normalize the features since this method computes the distance weighting each attribute equally. Thus, some attributes would have different importance when the distance is computed. For the k-means, it is proposed the normalization of the variable to take their values to the range 0-1, using Formula 3.1.

[3]Source: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html

After all clusters are formed, it is checked the stable cluster. It is counted the number of stable and non-stable frames that it has. Also, this count is repeated for the rest of clusters. In this way now, it is had the structure of the error for a binary problem of classification. The confusion matrix is shown in Table 3.1.

	Predicted class	
Actual class	**TP**: Stable frames in the stable cluster	**FN**: Non-stable frames in the stable cluster
	FP: Stable frames out of the stable cluster	**TN**: Non-stable frames out of the stable cluster

Table 3.1: Confusion matrix

With the structure of the error is possible to compute an estimation of the success rate of this model with the Formula 3.3, where P is the number of stable frames in the predictions and N is the non-stable frames:

$$Success\ rate = (\frac{TP + TN}{P + N}) * 100 \tag{3.3}$$

Chapter 4

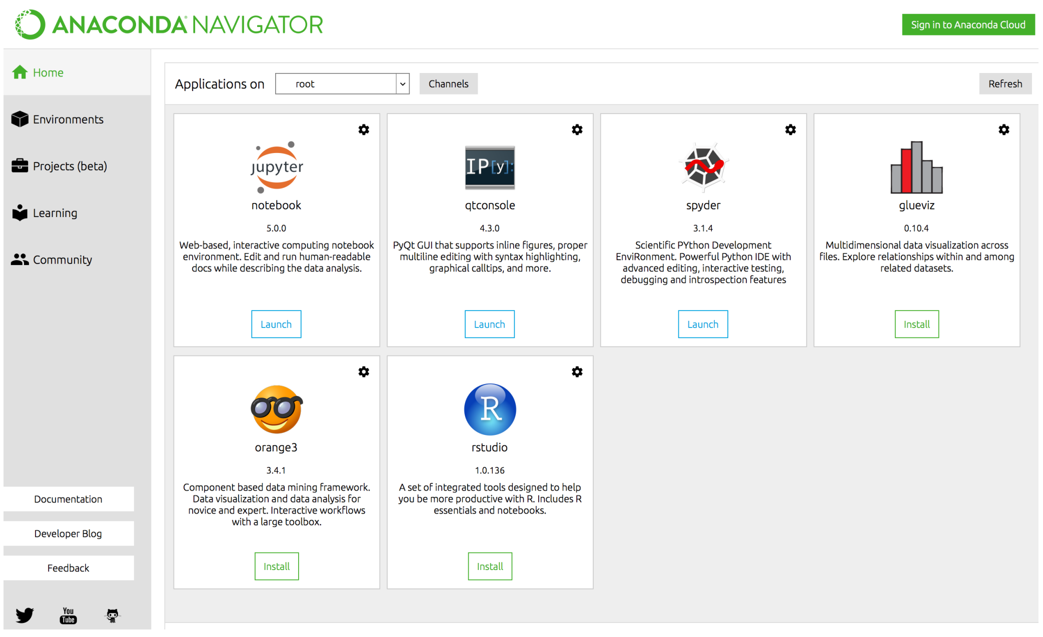
Implementation

The chapter about implementation includes information about how the system have been developed based on the proposal described in the previous chapter. It also explains the difficulties or challenges that have been faced to fulfill the most of the formalities of the proposal and, if they have been achieved with a success result.

4.1 Software

In this section is exposed the software and tools to carry out the implementation of the proposal.

The work environment has been included in Anaconda navigator (Figure 4.1) which is a desktop graphical user interface (GUI) that allows to launch applications and manage the packages in a easy way which can be used different version of the same package to take advantage of the features of each version, if it was the case.



Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

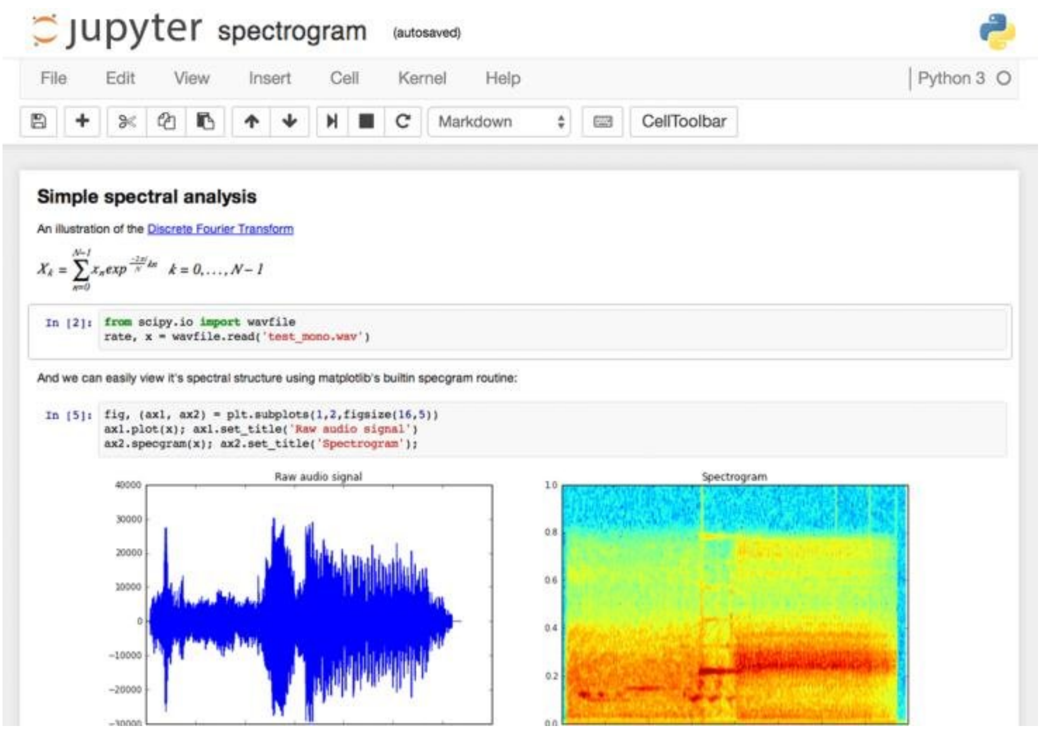
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Figure 4.1: Anaconda navigator

Inside Anaconda, the application of Jupyter Notebook (Figure 4.2) will save the code of the project. It is a web-based interactive computational environment to develop notebooks. These notebooks will be the document format structured as a JSON file and extension .ipynb, that will contain an ordered list of cells which can contain more than code as graphics, text for descriptions, mathematics or other complex outputs.


jupyter spectrogram (autosaved)
File Edit View Insert Cell Kernel Help
Python 3
Markdown
CellToolbar

Simple spectral analysis

An illustration of the Discrete Fourier Transform

$$X_k = \sum_{n=0}^{N-1} x_n exp^{\frac{-2\pi i}{N}kn} \quad k = 0, \ldots, N-1$$

```
In [2]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize(16,5))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram');
```

Figure 4.2: Jupyter notebook environment[1]

The programming language Python has been used to implement the different parts contained in the proposal:

- An annotated dataset which is the real class (stable or non-stable) of the frames.
- 4 feature sets related to the amount of motion and histogram of magnitudes through the extraction and computation of the modules of the optical flow vectors with both algortihms of Lucas Kanade and Gunnar Fanerbäck.
- The implementation of the classification system based on the 3 methods of threshold, support vector machine and k-means.

There are many advantages of using Python for this task: it provides a large toolbox for all kinds of applied programming problems, for example, related to the fields of computer vision and machine learning, and also offers a natural way of managing and manipulating data with the numpy library structuring data as n-dimensional arrays.

The library OpenCV (Open Source Computer Vision) contains a wide range of computer vision algorithms, from motion detection for security issues, to process control applications

[1]Source: https://www.researchgate.net/figure/Jupyter-notebook-for-python-26_fig3_327233649

where object recognition is required. In the case of this work, the purpose of using this library will be to compute the optical flow vectors given the video frames. From version 3.0.0, OpenCV was adapted to work with Python, avoiding the old C++ interface and make the task easier with the support of libraries such as NumPy with optimized numerical operations.

On the other hand, Scikit-learn will help to implement the classification methods. It is a library focuses on the field of machine learning and contain algorithms about classification, regression and clustering which are available to work with Python. Moreover, it is designed to work with numeric and scientific libraries such as NumPy and ScyPy.

4.2 Architecture

The core of the system's architecture is built mainly with two subsystems: the features extraction and the modeling and testing of classification systems as it can be seen in Figure 4.3.

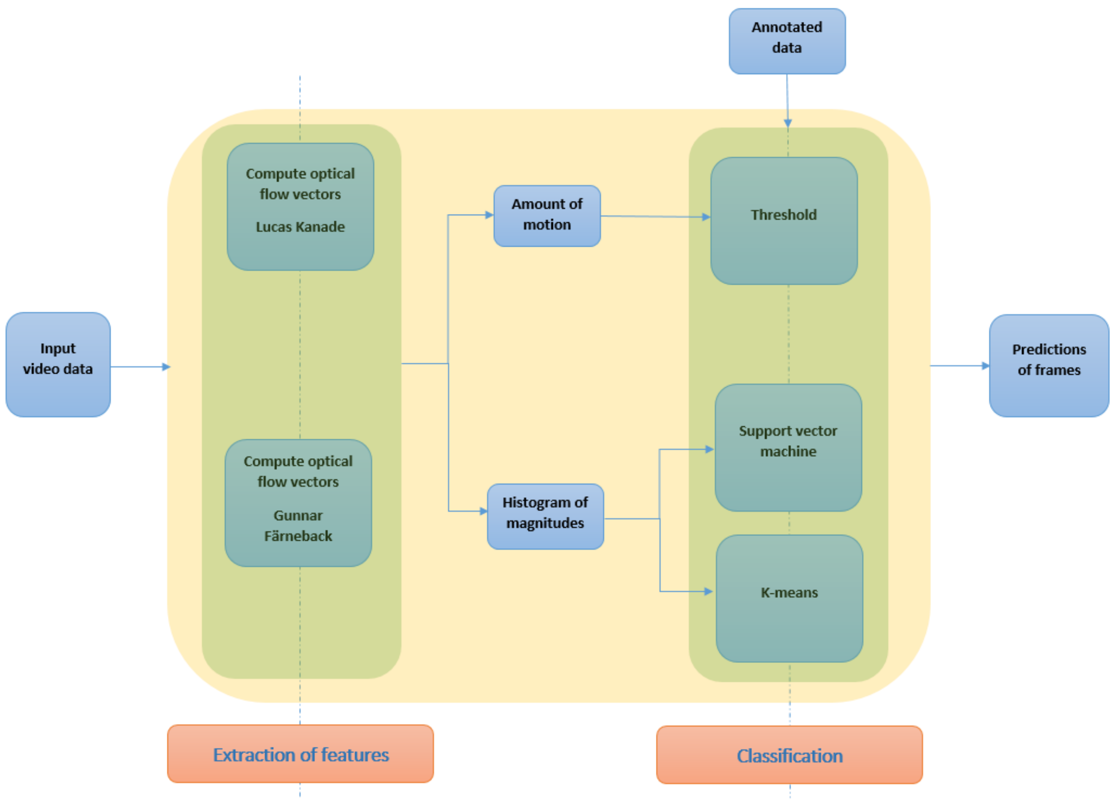


Figure 4.3: System's architecture

Each subsystem has their inputs and outputs. In the case of features extraction, it uses the sequence of video frames as input and produces the features of amount of motion and histogram of magnitudes as outputs, creating two set of features based on the computing of the optical flow vectors through sparse or dense approaches. In the case of classification, it uses the produced featuers from the features extraction subsystem, to be the inputs along with the dataset of video annotations to train and test the respective classification models.

The output of the system will be the predictions of the video frames from the optimal trained models.

4.3 Dataset of video annotations with HSV model

The annotations of video serve to train the supervised classification models with their convenient testing to assess the method (the same case for unsupervised approaches).

To make the annotations and get a quality dataset, it is made the decision of extracting a sequence of 3000 consecutive frames from the beginning of the video with the representation of computed optical flow vectors by a dense approach using color model HSV to have a whole perspective of the sequence.

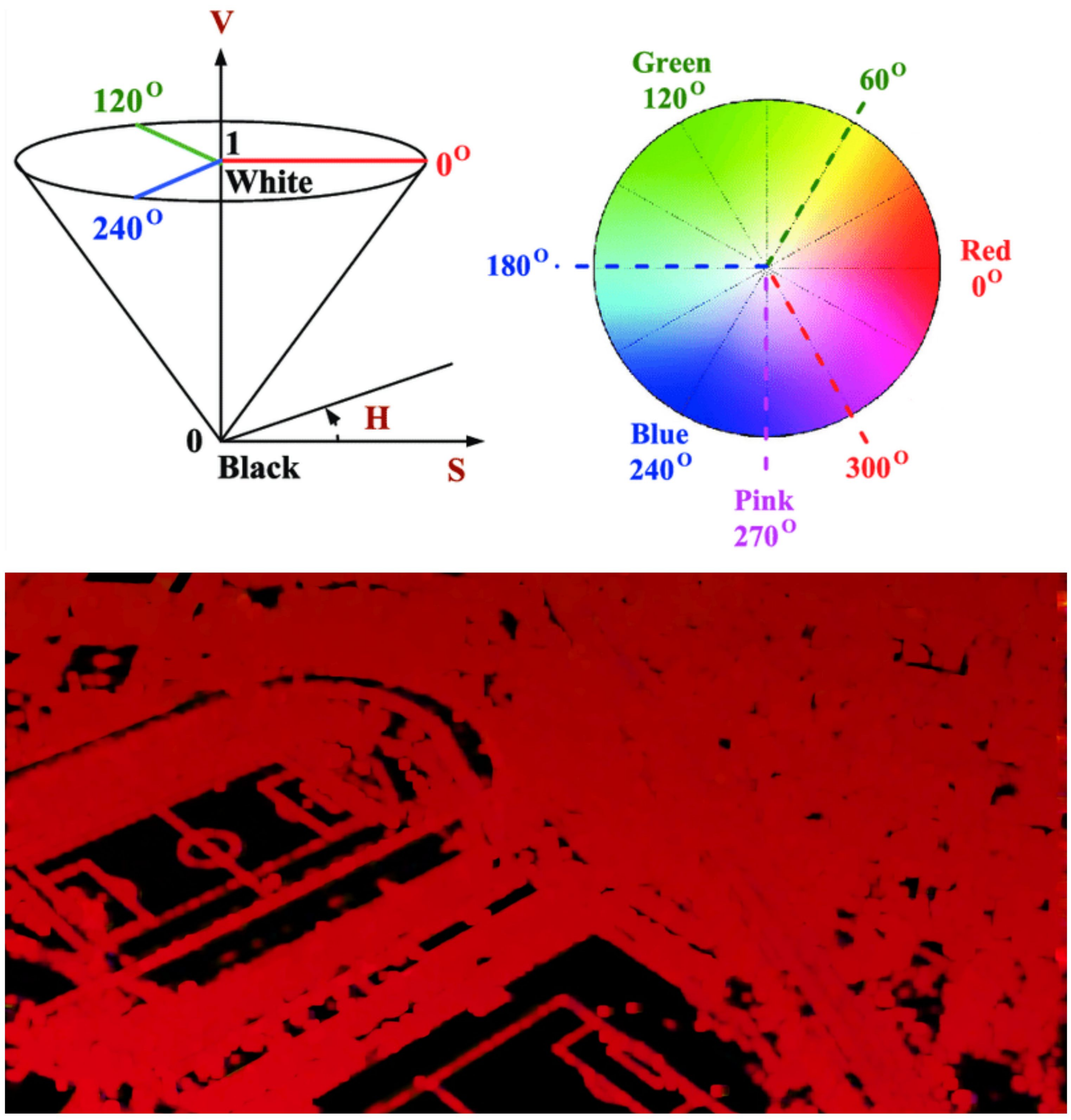


Figure 4.4: HSV representation[2]

As can be seen in Figure 4.4 is achieved a more visual way to distinguish the stable frames from the non-stable using the HSV model color to draw the optical flow vectors. If only it had been extracted the frames in their common RGB channels, and start to pass the frames to determine if there is enough movement to consider the frame stable or not, it is possible to misclassify the frame. It drives to a bad training and a creation of model that it will not work.

With the HSV model is possible to see the magnitude and direction of the optical flow vectors through the hue and value components of the color model. The saturation component is configured with maximum possible value. In the case of 4.4 the direction has 0^{o} and the module of the most optical flow vectors are large, consequently there is a hard movement for this frame.

4.4 Features extraction

In this section it is explained the implementation details about the extraction of the information with the computation of the optical flow vectors through Lucas Kanade and Gunner Farnebäck to get the features of amount of motion and histogram of magnitudes. For both methods it is used the respective functions contain in the library OpenCV 3.4.2.

Lucas Kanade

The main function of OpenCV to compute the optical flow vectors with a sparse approach is cv2.calcOpticalFlowPyrLK(), but it is also necessary to select from the whole set of frame's pixels, the potencial points which could be good options to track between the transition of two frames. At this point is where the method cv2.goodFeaturesToTrack() takes importance.

The method cv2.goodFeaturesToTrack() based on Shi-Tomasi detector, selects the N strongest corners of a grayscale image and return their location in the image. In general, the interest is to select a sample that contains a large number of points. The configuration of the parameters of the function are the following:

- maxCorners = 500
  - It is decided to select a sample of big size of points, where the algorithm will select the strongest 500 corners or less.
- qualityLevel = 0.2
  - A value between 0 and 1, the quality must not be too high since it will select less points.
- minDistance = 7
  - The euclidean distance that the returned points must be separated at least. It is interested a standard distance because if the distance is too high the size of points could be small, but if the distance is too small it could select lots of point from just a region of the frame when the goal is to have a general perspective of the movement in the complete frame.

[2]Source: https://www.researchgate.net/figure/HSV-color-space-and-RGB-color-transformation_fig4_312678134

- blockSize = 7
  - The size of the neighbourhood around each pixel to compute the derivative covariation matrix.
- image = multi-dimensional array with the same shape than the original frame, where each value represent the pixel with just a single color channel
  - The previous frame in grayscale from the one that will be computed the optical flow.

The method cv2.calcOpticalFlowPyrLK() computes the optical flow vectors of the given points by the corner detector and mixing with the approach of the pyramids: rescale the image going up in the pyramid, giving a result that when a image is smaller the registered movement will be smaller too, disappearing the smaller and staying the larger ones. The configuration of the parameters is the following:

- winSize = (15,15)
  - It is the dimension of the window's search in each pyramid level where it is tried to find the point in the next frame.
- maxlevel = 2
  - It is level of the pyramid where the search is started. In the case of the number of levels of the pyramid is 3.
- criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
  - The criteria to finish the search in each level of the pyramid. In this case is configured to stop the process when is reached the maximum number of iterations (10) or when the search window moves by less than epsilon (0.03), the minimum search window it can process on.
- prevImage = multi-dimensional array with the same shape than the original frame, where each value represent the pixel with just a single color channel
  - The previous frame in grayscale from the one that will be computed the optical flow.
- nextImage = multi-dimensional array with the same shape than the original frame, where each value represent the pixel with just a single color channel
  - The next frame in grayscale where will be done the search.
- prevPoints = multi-dimensional array with points that will be computed the optical flow vectors
  - It is the output of the function cv2.goodFeaturesToTrack().

This procedure returns the optical flow vectors in x,y components with a status for each point, indicating if the flow have been found or not. The points with a wrong computation of the flow will be removed.

It is used a loop to iterate over the sequence of frames and extract in each iteration as the amount of motion as the histogram of magnitudes. Also in each iteration is selected new points to track with the method cv2.goodFeaturesToTrack(), since the target is to get the maximum information between each two frames. If on the contrary, it would be let the same set of points along a sequence of frame, the features will suffer a loss of the information since the flow will more difficult to find and some of the previously selected points will be useless.

Gunnar Farnebäck

To carry out the computation of the optical flow with a dense approach, it is used the method cv2.calcOpticalFlowFarneback(). It will compute the optical flow vectors for the whole set of points given two frames applying the pyramid approach. The configuration of the parameters of the method is the following:

- pyr_scale = 0.5
    - The value is 0.5 to do a rescale of two times smaller the image when is going up to the next layer of the pyramid. The range can be between 0 and 1.
- levels = 3
    - It is the number of layers of the pyramid. In this case is used a pyramid of 3 levels counting the base or the original image.
- winsize = 15
    - It is the dimension of the window's search in each pyramid level where it is tried to find the point in the next frame.
- iterations = 3
    - Number of iterations in each layer of the pyramid.
- poly_n = 5
    - It determines the size of the neighbourhood to find a polynomial expansion in each pixel.
- poly_sigma = 1.2
    - The value indicates the standard deviation of the Gaussian. It will smooth the derivatives used as a basis for the polynomial expansion.
- prev = multi-dimensional array with the same shape than the original frame, where each value represent the pixel with just a single color channel

    - The previous frame in grayscale from the one that will be computed the optical flow.

- next = multi-dimensional array with the same shape than the original frame, where each value represent the pixel with just a single color channel

    - The next frame in grayscale where will be done the search.

The output of this procedure will be the displacement of each pixel with respect the previous frame, given with x,y components. It is used a loop to iterate over the sequence of the frames and compute the optical flow between each two frames.

Amount of motion and histogram of magnitudes

After the computation of the optical flow vectors with both methods, it is time to create two feature sets for each approach: amount of motion and histogram of magnitudes.

For the amount of motion, it will be calculated the modules of the optical vectors with the locations and displacements that return each method. It will be used the euclidean norm to compute the module of each vector and then compute the sum of them, to reach to the value of amount of motion for a determined frame.

In the case of histogram of magnitudes, first it will be computed the magnitudes through the modules of the optical flow vectors using the euclidean norm and then it will count how many of them has a length from 0 to 5, from 5 to 10, ans so on, until how many have a length of 50 or more. Each bin of the histogram will be the frequency of the vectors with a determined length in a concrete frame.

4.5 Training and testing the classification methods

This section is about the implementation details of the process of training classification models to get the optimum from each approach to get a proper classification of the frames depending on the amount of motion or the histogram of magnitudes. It is used methods from the libraries OpenCV 3.4.2 and Scikit-learn 0.20.3.

10-fold cross validation

This technique is implemented as to be used in support vector machine as in the threshold methods. It will help to determine the skill of the classification procedure with new frames that are unknown if they are stable or non-stable, and it is just know its amount of motion or histogram of magnitudes.

Working with annotated data with the normalized values of amount of motion or histogram of magnitudes, *my_data_norm*, it is splitted by its category into tow sets: stables frames and non-stable frames. This step is to be able to stratify the sets of training and testing.

```
stable = my_data_norm[my_data_norm[:,-1].ravel()==1]
unstable = my_data_norm[my_data_norm[:,-1].ravel()==0]
```

Then, using a loop of 10 iterations both sets are splitted using the a set of indexes that they will be the 10% of the size of each stable and unstable set to build the test set joining the two parts. The same case for the training set using the rest of examples. In this manner, in the first iteration is taken the first 10% of the examples for each category, in the second iteration is taken the second 10% of the example of each category, and so on.

```
for i in range(10):
    inf = int(i*0.1*stable.size)
    sup = int((i+1)*0.1*stable.size)

    temp = np.split(stable,[inf,sup],axis=0)
    test_stable = temp[1]
    train_stable = np.concatenate((temp[0],temp[2]))

    inf = int(i*0.1*unstable.size)
    sup = int((i+1)*0.1*unstable.size)

    temp2 = np.split(unstable,[inf,sup],axis=0)
    test_unstable = temp2[1]
    train_unstable = np.concatenate((temp2[0],temp2[2]))
```

Threshold

In this classification method is searched the threshold that separates in the best way the frames from the stable group then the non-stable group based on the normalized amount of motion.

Using the 10-fold cross validation to see which threshold works better for each partition of the data in train and test sets, it is searched the threshold in the interval 0 to 0.15 as can be seen in Figure 3.7, where the optimal threshold must be. This decision reduces the time processing.

Thus, for each iteration in the cross the validation there is a search in the set of possible thresholds from 0 to 0.15 each 0.01, called the variable *threshold*. Fixing this number is possible to separate the train data into the stable group (1) or the non-stable group (0). The threshold with the maximum success rate for the train test will be the chosen one to check it with the test set.

```
for j in threshold:
    aux = np.where(train_stable <= j, 1, 0)
    aux2 = np.where(train_unstable > j, 1, 0)

    # Success rate train set
    calc = (sum(aux)+sum(aux2))/(aux.size+aux2.size)
    if success < calc:
        success = calc
        thres = j

# Success rate test set
aux_test = np.where(test_stable <= thres, 1, 0)
aux2_test = np.where(test_unstable > thres, 1, 0)
success_test = (sum(aux_test)+sum(aux2_test))/(aux_test.size+aux2_test.size)
```

Support vector machine

To generate a classification model based on the support vector machine it is used the function svm.SVC().fit() from the library Scikit-learn. The configuration of the parameters for SVC is the following:

- kernel = 'linear', 'sigmoid', 'rbf'
    - It is used 3 different types of kernels to check what of them is adapted better to these type of data.
- gamma = 'auto'
    - It is adjusted the gamma value for the sigmoid and rbf kernels.

In the case of fit() is included the training set in the first argument and class variable in the second argument.

The method will return a SVC object which contains the method predict() to make the predictions of the test set that must be included as argument.

K-means

For the unsupervised approach, the K-means model has been implemented using the function cv2.kmeans() from the library OpenCV. The chosen configuration of the parameters is:

- samples = histogram of magnitudes
    - The used features to group the examples in k clusters.
- k = 5, 10, 25
    - It will be used different numbers of clusters to group the data and see which works better.
- criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
    - It is chosen a termination criteria for the algorithm based on reaching the epsilon value (1) or the maximum number of iterations (10).
- attempts = 10
    - This is the number of times choosing between different initial labelling.
- flags = cv2.KMEANS_RANDOM_CENTERS
    - It is chosen the option of taking k random centers at the beginning of the algorithm.

The method cv2.kmeans() returns the labels of each example contains in the argument 'samples'. Also it returns the locations of the centers of the cluster and a dispersion measure for each cluster.

After grouping the examples in k clusters, it has to discover the cluster the largest number of stable frames, to denominate it the stable cluster. The other clusters will be considered from the non-stable group.

To make new predictions, it takes a new example and it will be associated to the closest cluster in euclidean distance.

Chapter 5

Testing and results

In this chapter it is evaluated the adopted approach in this work to classify frames into frames with no harsh movement or non-stable. It will be exposed the results produce by the performance of different classification methods and their skill to fulfill the task of separating the frames in these two groups based on the inputs of amount of motion or histogram of magnitudes.

It will be seen the success rate that they get with the prepared feature set and the different configuration of each model.

5.1 Results applying a threshold with amount of motion

The evaluation of this method is carried out through 10-fold cross validation applying to the generated feature set of amount of motion processed with Lucas kanade and Gunnar Färneback. Moreover, it is used as input the dataset of annotations to discover the real class of each frame.

In summary, it is dealt with a sample of 3000 frames to train and test the classification method with these following files:

- 3000 examples of amount of motion computed by a sparse method.
- 3000 examples of amount of motion computed by a dense method.
- 3000 examples of annotations dataset with the class of each frame.

The algorithm uses the data from amount of motion as for sparse approach as dense, and it is normalized by columns, applying formula 3.1.

The normalized data is shuffled to avoid problems in the classification process in the case that the examples have been ordered by some criteria.

The results obtained by the algorithm for the sparse approach can be seen in Tables 5.1 with the success rate and the optimal threshold obtained for each fold in the cross validation procedure.

In table 5.2 is found the optimal threshold as the average of all threshold of the ten folds and the final success rate of the algorithm with these data.

Fold	Threshold	Success rate
1	0.09	0.943
2	0.09	0.937
3	0.09	0.933
4	0.09	0.933
5	0.09	0.953
6	0.09	0.920
7	0.09	0.927
8	0.09	0.930
9	0.09	0.933
10	0.09	0.924

Table 5.1: Threshold and success rate with a sparse approach

	Threshold	Success rate
Average	0.09	0.933

Table 5.2: Optimal threshold and final success rate with a sparse approach

The results show a good performance of the algorithm classifying correctly the 93.3% of the generated examples with the algorithm of Lucas Kanade based on the amount of motion. Also in all folds the value 0.09 of the threshold is the optimum to separate the normalized amount of motion in stables or non-stables frames.

The tables 5.3 and 5.4 contain the results for the amount of motion processed by Gunnar Farnebäck.

Fold	Threshold	Success rate
1	0.09	0.983
2	0.09	0.987
3	0.09	0.973
4	0.09	0.970
5	0.09	0.967
6	0.09	0.980
7	0.09	0.970
8	0.09	0.987
9	0.09	0.980
10	0.09	0.967

Table 5.3: Threshold and success rate with a dense approach

	Threshold	Success rate
Average	0.09	0.976

Table 5.4: Optimal threshold and final success rate with a dense approach

The results are improved respect the amount of motion processed by Lucas Kanade in a little more of 4 percentage points but maintaining the same optimal value 0.09 for the threshold along the different folds.

Visually is possible to see in Figure 5.1 how the dense approach separates better the examples than the sparse, producing a more compacted amount of motion for the stable group being easier to discriminate both groups.

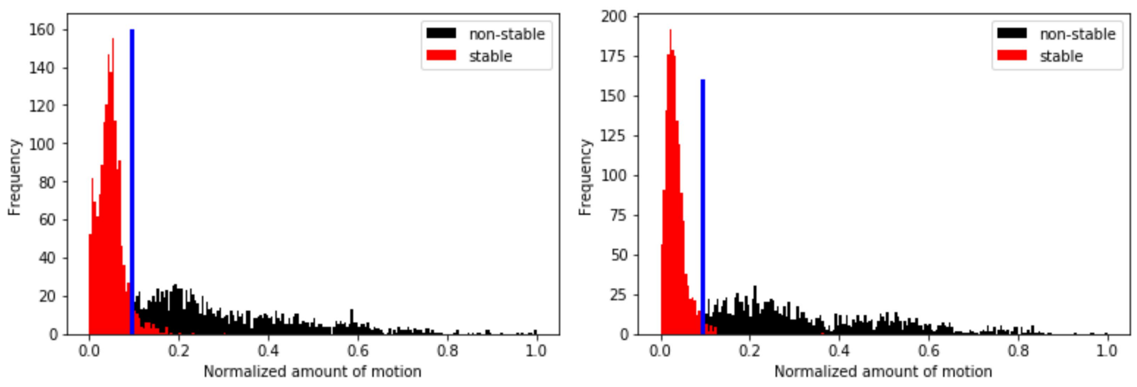


Figure 5.1: Threshold in amount of motion with sparse (left) and dense (right)

5.2 Performance of support vector machine with histogram of magnitudes

The support vector machine classification is assessed using the 10-fold cross validation, and unlike the threshold classification method it will use the feature set of histogram of magnitudes based on both methods of Lucas Kanade and Gunnar Farnebäck. Thus, this method will have more attributes to determine the classification. The annotations dataset will be necessary to check the success in the predictions.

In summary, it is dealt with a sample of 3000 frames to train and test the support vector machine classification method with these following files:

- 3000 examples of histogram of magnitudes computed by a sparse method.
- 3000 examples of histogram of magnitudes computed by a dense method.
- 3000 examples of annotations dataset with the class of each frame.

The feature set of histogram of magnitudes is normalized applying the Formula 3.1, and then is shuffled for the same reasons as in the threshold method.

The support vector machine method is used with different types of kernel to observe if differences exist among them. The results obtained by the algorithm for the sparse approach can be seen in Tables 5.5 with the success rate for each kernel obtained in each fold in the cross validation procedure.

In table 5.6 is found the final success rate of the algorithm with these data.

Fold	Success rate		
	Linear kernel	Radial Basis kernel	Sigmoid kernel
1	0.960	0.973	0.960
2	0.977	0.973	0.963
3	0.973	0.960	0.963
4	0.963	0.960	0.960
5	0.977	0.947	0.957
6	0.963	0.983	0.947
7	0.963	0.963	0.953
8	0.973	0.963	0.947
9	0.987	0.950	0.967
10	0.963	0.967	0.977

Table 5.5: Success rate for each kernel with a sparse approach

	Success rate		
	Linear kernel	Radial Basis kernel	Sigmoid kernel
Average	0.970	0.964	0.959

Table 5.6: Final success rate for each kernel with a sparse approach

In the sparse case, it gets a high percentage of good classification frames in the three cases, all above 95%. The differences among them are small, but using the linear kernel, it is possible to get a 97% of success in good classified instances, the highest one.

The tables 5.7 and 5.8 contain the results for the histogram of magnitudes processed by Gunnar Farnebäck.

Fold	Success rate		
	Linear kernel	Radial Basis kernel	Sigmoid kernel
1	0.977	0.983	0.967
2	0.973	0.980	0.980
3	0.990	0.987	0.977
4	0.973	0.993	0.990
5	0.987	0.977	0.963
6	0.993	0.963	0.993
7	0.963	0.990	0.983
8	0.990	0.983	0.970
9	0.983	0.967	0.987
10	0.967	0.970	0.983

Table 5.7: Success rate for each kernel with a dense approach

	Success rate		
	Linear kernel	Radial Basis kernel	Sigmoid kernel
Average	0.980	0.979	0.979

Table 5.8: Final success rate for each kernel with a dense approach

The obtained results with the feature set of dense histogram of magnitudes, are better in the three cases than the obtained results with the sparse histogram of magnitudes. They have improved in one percentage point in the case of linear kernel, one and a half percentage point with radial basis kernel and two using the sigmoid kernel. Moreover, now the differences among the three are smaller and the most successful kernel is the linear with a 98% of good classified instances.

5.3 Analysis of results using K-means for the histogram of magnitudes

In this section is evaluated the unsupervised method of k-means to see its skill to organize in clusters the different frames based on the sparse and dense histogram of magnitudes.

The classification task is composed of different steps to compute the success rate for the method:

1. It is used a different number of clusters to execute the algorithm. It is chosen the numbers 5, 10 and 25 to see the variation in the success rate. It is wanted to check the assumptions that inside the stable group, there are not subgroups, unlike the non-stable group.
2. After the execution of the algorithm is seen which is the cluster with the largest number of stable frames, to denominate it as the stable cluster. The rest will be considered as non-stable and each one will be a subgroup: non-stable due to translation, rotation, zooming, or a different phenomenon that could cause a high motion in the frame.
3. To compute the success rate is necessary to compute the confusion matrix:
    - TP: Stable frames in the stable cluster
    - FP: Unstable frames in the stable cluster
    - FN: Stable frames outside the stable cluster
    - TN: Unstable frames outside the stable cluster
4. The success rate will be the sum of the diagonal divided by the sum of the whole matrix.

The input files used for the algorithm are the following:

- 3000 examples of histogram of magnitudes computed by a sparse method.
- 3000 examples of histogram of magnitudes computed by a dense method.
- 3000 examples of annotations dataset with the class of each frame.

At the beginning, the feature set of histogram of magnitudes is normalized applying the Formula 3.1.

K-means is sensible to the initial centers producing different results each time is executed (its implementation is with the random election of the centers). Thus, it will be run 10 times to get 10 success rates and to be able to get an average of them.

The table 5.9 includes measures for the sparse histogram of magnitudes the success rate (SR), the rate of stable frames inside the stable cluster about its size (SFSC) and the rate of stable frames in the stable cluster about the distribution over other clusters (SF). The table also divided the results by the number of generated clusters.

The table 5.10 contains the average of the 10 iterations of the algorithm about the different measures in each number of clusters (5,10 or 25).

	5 clusters			**10 clusters**			**25 clusters**		
Iteration	**SR**	**SFSC**	**SF**	**SR**	**SFSC**	**SF**	**SR**	**SFSC**	**SF**
1	0.913	0.949	0.887	0.868	0.953	0.796	0.812	0.989	0.659
2	0.843	0.951	0.748	0.846	0.927	0.775	0.944	0.961	0.935
3	0.948	0.941	0.965	0.942	0.944	0.949	0.875	0.952	0.809
4	0.947	0.948	0.955	0.835	0.977	0.711	0.870	0.959	0.793
5	0.844	0.959	0.743	0.879	0.947	0.823	0.846	0.961	0.745
6	0.853	0.970	0.751	0.943	0.934	0.962	0.722	0.991	0.489 (0.271)
7	0.922	0.947	0.907	0.760	0.986	0.564 (0.389)	0.933	0.930	0.947
8	0.943	0.945	0.949	0.857	0.963	0.765	0.797	0.978	0.639 (0.345)
9	0.813	0.972	0.673 (0.230)	0.849	0.950	0.761	0.803	0.983	0.646
10	0.816	0.965	0.684 (0.299)	0.857	0.958	0.769	0.876	0.979	0.788

Table 5.9: SR, SFSC and SF for each number of clusters for sparse histogram of magnitudes

	5 clusters			**10 clusters**			**25 clusters**		
	SR	**SFSC**	**SF**	**SR**	**SFSC**	**SF**	**SR**	**SFSC**	**SF**
Average	0.884	0.955	0.8262	0.864	0.954	0.788	0.848	0.968	0.745

Table 5.10: Average SR, SFSC and SF for each number of clusters for sparse histogram of magnitudes

The results show a pretty good success rate for each number of clusters, but better in the case of a low number of clusters, in this case 5 clusters. The success rate is related to the SF since with a low rate of stable frames inside the stable cluster about the others, it causes a increase of misclassified of stable frames. In the SFSC value is possible to see that the size of the considered stable cluster contains at least a 95% of stable frames for the three number of clusters.

In the case of sparse histogram of magnitudes there are values in the SF with a low rate. For example, in iteration number 6, the value is 0.489, that it means that from the whole stable frames are just the 48.9% are in the stable cluster. That is because in these cases there is another cluster with a significant rate of stable frames which is considered as non-stable, and its SF is also indicated inside the parenthesis (0.271) after the value for stable cluster. Thus, there are some cases using sparse histogram of magnitudes where the assumption of no subgroups in stable group fails.

The tables 5.11 and 5.12 includes the values for the measures SR, SFSC and SF of each number of clusters in the case of the dense histogram of magnitudes.

	5 clusters			10 clusters			25 clusters		
Iteration	**SR**	**SFSC**	**SF**	**SR**	**SFSC**	**SF**	**SR**	**SFSC**	**SF**
1	0.940	0.900	0.999	0.939	0.899	0.999	0.909	0.856	0.999
2	0.941	0.902	0.999	0.904	0.999	0.824	0.958	0.930	0.998
3	0.955	0.955	0.920	0.956	0.927	0.997	0.981	0.988	0.977
4	0.960	0.938	0.992	0.939	0.902	0.996	0.972	0.958	0.991
5	0.960	0.938	0.992	0.980	0.980	0.983	0.972	0.960	0.989
6	0.964	0.994	0.939	0.938	0.900	0.996	0.979	0.977	0.985
7	0.956	0.995	0.924	0.951	0.919	0.997	0.948	0.914	0.999
8	0.945	0.908	0.999	0.977	0.970	0.988	0.98	0.983	0.980
9	0.981	0.988	0.977	0.951	0.921	0.994	0.916	0.869	0.994
10	0.916	0.865	0.999	0.938	0.898	0.999	0.949	0.914	0.999

Table 5.11: SR, SFSC and SF for each number of clusters for dense histogram of magnitudes

	5 clusters			10 clusters			25 clusters		
	SR	**SFSC**	**SF**	**SR**	**SFSC**	**SF**	**SR**	**SFSC**	**SF**
Average	0.952	0.938	0.974	0.947	0.932	0.977	0.956	0.935	0.991

Table 5.12: Average SR, SFSC and SF for each number of clusters for dense histogram of magnitudes

Using the dense histogram of magnitudes it is achieved much better results than with the sparse approach. Moreover, the differences using different number of clusters are more insignificant for the success rate with an approximate value of 0.95 for the three number of clusters. This happens because there is a high rate in SF, being at least the 97% of the whole set of stable frames in the stable cluster for the three cases (and then fulfilled the assumption of no subgroup in the stable group), containing almost 94% of its size just stable frames.

The only disadvantage about the dense histogram of magnitudes is that it is produced more misclassified unstable frames as stable since the SFSC is lower and the stable cluster usually contains more unstable frames for the dense histogram of magnitudes.

5.4 Comparison of the obtained results

In this section is shown the final results where is compared the performance of the different classification methods used in this work.

It is organised by 3 classification methods based on a sample of 3000 frames in sequence which are processed their optical flow vectors by Lucas Kanade and Gunnar Farnebäck, and grouped by two types of features set: amount of motion for the threshold method and histogram of magnitudes for the support vector machine and k-means. For each one is achieved the success rate for the sample.

The summary of results is shown in Table 5.13 which offers a broader perspective of the performance.

		Sparse	Dense
Threshold		0.933	0.976
SVM	**Linear kernel**	0.970	0.980
	Radial Basis kernel	0.964	0.979
	Sigmoid Kernel	0.959	0.979
K-means	**5 clusters**	0.884	0.952
	10 clusters	0.864	0.947
	25 clusters	0.848	0.956

Table 5.13: Success rate for each classification method

The first appreciable characteristic of the results is the difference among using a sparse approach or a dense to compute the optical flow vectors. In the second case, it is achieved a better performance in the success rate for all methods in the seven configurations shown by Table 5.13. The greatest differences come from the k-means method, where in the case of k-means with 25 centers the results are improved in almost 11 percentage points.

This worst realization of the sparse could be produced by a low quantity of extracted optical flow vectors, since with Gunnar Farnebäck is processed one for each pixel of the frame.

Comparing the three classification methods in general the unsupervised approach of k-means works worst than the other two. Moreover, support vector machine is which offers a better success rate in the classification process using a linear kernel. The threshold method is near the support vector machine but has a lower rate using the dense amount of motion as input with 97.6% of good classified frames.

Chapter 6

Conclusion

The aim of this work was to classify the frames of a video as stable or non-stable based on if they contained no harsh movement. To measure the motion of the video, it was generated features (amount of motion and histogram of magnitudes) from the processed video with sparse and dense optical flow approaches. Finally, the extracted features were classified and a high success rate was achieved in the predictions reaching the 98% of good classified instances using dense features with support vector machine. Thus, the expectations about the goal of this work are fulfilled.

It was checked that in general, process the datasets with sparse algorithms (Lucas Kanade) does not generate features as good as dense (Gunnar Farnebäck), lowering the rate of good classified frames significantly when they are used in the classification methods. On the other hand, they save time processing and, although the results are not the optimal, they are pretty good.

About this work, it is possible to continue with further work as serving this work as a first part in the video stabilization process, detecting the video frames which have some problem of stability and try to apply methods to fix and smooth the contained motion in those sequence of video.

Also, as had been seen when it was applied the K-means algorithm with different number of clusters, the non-stables frames were distributed along different clusters, which they were not the stable cluster. So that, it is suggested to discover which type of non-stability they are: translation, rotation, zooming; to model better the motion of the video and then apply the optimal video stabilization for that type of motion.

Bibliography

[1] Bhattacharyya, S.: *Support Vector Machine: Kernel Trick; Mercer's Theorem*. 2018. Retrieved from: https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d

[2] Downs, G. M.; Barnard, J. M.: *Hierarchical and non-Hierarchical Clustering*. 1995. Retrieved from: http://www.daylight.com/meetings/mug96/barnard/E-MUG95.html

[3] Farnebäck, G.: *Two-Frame Motion Estimation Based on Polynomial Expansion*. 2003. Retrieved from: http://www.diva-portal.org/smash/get/diva2:273847/FULLTEXT01.pdf

[4] Harris, C.; Stephens, M.: *A combined corner and edge detector*. 1988. Retrieved from: https://www.cis.rit.edu/~cnspci/references/dip/feature_extraction/harris1988.pdf

[5] Hastie, T.; Tibshirani, R.; Friedman, J.: *The Elements of Statistical Learning: data mining, inference and prediction*. chapter 12–14. Springer. second edition. 2009. ISBN 0387848843.

[6] Li, Y.; Zhang, X.; Chen, D.: *CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes*. 2018. Retrieved from: https://arxiv.org/pdf/1802.10062.pdf

[7] Lucas, B. D.; Kanade, T.: *An iterative image registration technique with an application to stereo vision*. 1981. Retrieved from: http://cseweb.ucsd.edu/classes/sp02/cse252/lucaskanade81.pdf

[8] Mahalingam, G.; Kambhamettu, C.; Aguirre, B.: *Crowd Motion Analysis from Video* . 2009. Retrieved from: https://pdfs.semanticscholar.org/d7f3/9548a126709fb45d9fc3b7c555e97e3779aa.pdf

[9] Mitchell, T. M.: *Machine Learning*. chapter 1. McGraw-Hill Science/Engineering/Math. 1997. ISBN 0070428077.

[10] O'Donovan, P.: *Optical Flow: Techniques and Applications*. 2005. Retrieved from: https://www.dgp.toronto.edu/~donovan/stabilization/opticalflow.pdf

[11] Rawat, P.; Singhai, J.: *Review of Motion Estimation and Video Stabilization techniques For hand held mobile video*. 2011.
Retrieved from: http://aircconline.com/sipij/V2N2/2211sipij13.pdf

[12] Sharma, P.: *Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques*. 2019.
Retrieved from: https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/

[13] Shi, J.; Tomasi, C.: *Good Features to Track*. 1994.
Retrieved from:
https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=323794

[14] Wikipedia: *Cross-validation (Statistics)*. 2003.
Retrieved from: https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation

[15] Wikipedia: *Machine Learning — Wikipedia, The Free Encyclopedia*. 2003.
Retrieved from: https://en.wikipedia.org/wiki/Machine_learning#Approaches