Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROJECTE FINAL DE CARRERA

Video Clustering Using Camera Motion

Studies: Telecommunication Engineering

Author: Laura Tort Alsina

Advisors: Michelle Rombaut, Denis Pellerin and Xavier Giró-i-Nieto

Year: 2012

Resum del Projecte

Aquest document recull el treball fet al INP Grenoble durant el segon semestre del curs 2011-2012, completat a Barcelona durant els primers mesos del curs 2012-2013. El treball presentat consisteix en un estudi del moviment de càmera en diferents tipus de vídeo per a agrupar fragments que tinguin certa similitud en el contingut.

En el document s'explica com es tracten les dades extretes pel programa *Motion 2D*, proporcionat per la universitat francesa, per tal de simplificar-ne la representació mitjançant histogrames de moviment. També s'explica com es calculen les diferents distàncies entre aquests histogrames i com es computa la seva similitud.

Es fan servir tres distàncies diferents: Manhattan, Euclideana i Bhattacharyya, tot i que en el marc del treball se n'ha explicat algunes de més complicades. També es fan servir diferents configuracions d'histogrames de moviment, fent servir més o menys contenidors per a representar el moviment.

Totes les possibles combinacions de número de contenidors i distàncies són avaluades fent servir un conjunt de 30 fragments de vídeo i l'algoritme de clustering K-Means. Els resultats del clustering s'avaluen fent servir el F_1-Score, una mesura molt popular que serveix tant per a algoritmes d'agrupament com per als de classificació.

Paraules clau: Agrupació de vídeo, Càmeres cinematogràfiques, Càmeres de televisió, Càmeres de vídeo, Característiques de moviment, Classificació, Clips de vídeo, Clústers, Desplaçament, Distàncies, Histograma, Moviment, Recuperació de la informació, Similitud i Vídeo.

Resumen del Proyecto

Este documento recoge el trabajo hecho en el INP Grenoble durante el curso 2011-2012, completado en Barcelona durante los primeros meses del curso 2012-2013. El trabajo presentado consiste en un estudio del movimiento de cámara en diferentes tipos de vídeo para agrupar fragmentos que tengan cierta similitud en el contenido.

En el documento se explica cómo se tratan los datos extraídos por el programa *Motion 2D*, proporcionado por la universidad francesa, con tal de simplificar su representación mediante histogramas de movimiento. También se explica cómo se calculan las diferentes distancias entre histogramas y como se computa su similitud.

Se usan tres distancias diferentes: Manhattan, Euclidiana y Bhattacharyya, aunque en el marco del trabajo se han explicado algunas un poco más complejas. También se utilizan diferentes configuraciones de histogramas de movimiento, utilizando más o menos contenedores para representar el movimiento.

Todas las posibles combinaciones de número de contenedores y distancias son evaluadas utilizando un conjunto de 30 fragmentos de vídeo y el algoritmo de clustering K-Means. Los resultados del clustering se evalúan utilizando el F_1-Score, una medida muy popular que sirve tanto para algoritmos de agrupación como para los de clasificación.

Palabras clave: Agrupación de vídeo, Cámaras cinematográficas, Cámaras de televisión, Cámaras de vídeo, Características de movimiento, Clasificación, Clips de vídeo, Clústers, Desplazamiento, Distancias, Histograma, Movimiento, Recuperación de la información, Similitud, Vídeo.

Abstract

This document contains the work done in INP Grenoble during the second semester of the academic year 2011-2012, completed in Barcelona during the firsts months of the 2012-2013. The work presented consists in a camera motion study in different types of video in order to group fragments that have some similarity in the content.

In the document it is explained how the data extracted by the program *Motion 2D*, proportionated by the French university, are treated in order to represented them in a more simplified using motion histograms. It is also explained how the different distances between histograms are calculated and how its similarity is computed.

Three different distances are used: Manhattan, Euclidean and Bhattacharyya, although in the project there can be found the explanation of some others a little bit more complicated. Different histogram configurations are used, using more or less bins to represent the motion.

Every possible combination of the number of bins and distances are evaluated using a group of 30 fragments of video and the clustering algorithm K-Means. The clustering results are evaluated using F_1-Score, a very popular measurement suitable for clustering algorithms and also classification.

Keywords: Cinematographic camera, Classification, Clusters, Distances, Histogram, Information retrieval, Motion, Motion features, Movement, Similarity, Television camera, Video, Video camera, Video clips and Video grouping.

Acknowledgements

This project could not have been possible without the three directors I have had: Michelle Rombaut and Denis Pellerin in the *Institut National Polytechnique de Grenoble* (Grenoble INP) and Xavier Giró-i-Nieto, in *Universitat Politècnica de Catalunya*, Barcelona. But there are much more people who I should be thankful to.

To begin with, Ferran Marquès, the current director of the *Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona* (ETSETB) and the teacher who gave me advice to help me choose where to do my second Eramus. Together with him, Alice Caplier, his contact in Grenoble INP, so I could choose a project that really enjoyed.

"Mes colocataires" in Grenoble: Ahmed, Sabine, Marièke and Isandra were like my family, asking me about the project and even offering to rehearse for the presentation I did in Grenoble.

My friends at home, as well as my partner, they have understood all the times I had to cancel plans because I was writing, and have never complained. And not to forget my colleagues, explaining anecdotes from their own projects so I was less worried.

Finally, my mother has been the most understanding person of all the list. She has been worried every single day until I have delivered the document. Doing everything I needed to be done: from a lunchbox to picking me at the bus station at 6 six in the morning when I came to visit from Grenoble.

To all this people, thank you for being there in the most important challenge of my university life.

Table of contents

1. Introduction

This document is a first approach of video clustering using camera motion. Video clustering is defined as grouping those video assets according to certain similarity criterion. In this work, this feature is motion. Nevertheless, this thesis originally proposed to be added in a multi-feature clustering algorithm that would analyse other features, such as dominant colour, orientation and sound.

There are several applications for video clustering, like assisting the user during annotation by expanding a label among all the elements in the cluster, or browsing through large video datasets by representing each cluster with a single icon in the user interface.

In this work, two videos are considered similar if they show similar situations, not the same elements or the same place, but the same action or event: people walking, a person getting out of a car, a door closing… These semantic concepts that are very easy to detect by a human being are not so easy for a computer program, so there are a lot of different approaches to that problem. In all those systems, the goal is modelising what is happening in the image and the changes suffered in the course of the sequence.

When working with video there are two types of features to be taken into account: the features that could be extracted from one single image, such as colour or texture; but also some temporal features that are exclusive for videos. Furthermore, there are some techniques that only analyse parts of the images: relevant points or regions, and their temporal evolution. In this project all the image is taken into account, and the sequence is analysed globally, making the most of all the dynamic information that can be extracted.

To simplify the study, the videos under analysis correspond to shots that have been previously extracted manually, the complete system would require an additional tool that could detect the cuts in a video and extract each shot so they could be analysed independently.

The technique used in this project measures the camera motion by using a previous work from *INRIA*[1] that extracts the dominant motion of the sequence, which is usually generated by

[1] INRIA (Institut National de Recherche en Informatique et en Automatique) is a públic research institution in France focusing in computer science.

the motion of the camera. These features are used to build a histogram of motion, and then each histogram is compared with the other videos to create clusters of similar videos.

This project has explored the possibilities of an existing motion features extractor provided by the GIPSA Lab. The parameters of the features extractor were studied to select the most relevant ones in order to obtain results that match as much as possible the human interpretation of videos. That is the reason why some features were discarded, as the objective was to obtain a clustering similar to what a human would have done with the same information.

Finally, it is important to point out that all the work presented in this document has been done in two stages, it was started in INP Grenoble in France and it has been finished in UPC, in Barcelona.

In Grenoble, Professors Michelle Rombaud and Denis Pellerin from the GIPSA-lab (Grenoble Image Parole Signal Automatique Laboratoire) provided the feature extractor as well as the main idea of the project. When the six months at Grenoble finished, I decided to continue the experiments that could not be finished in order to have a more complete knowledge of the problem and provide better conclusions. For this reason, a second period of thesis was performed at the Image Processing Group at UPC, with Professor Xavier Giró i Nieto.

2. Requirements

The main objective of this project was finding a simple way of analysing video shots and extracting information about its motion in order to make the annotation of videos easier. The project had already been started by a former student in the GIPSA-lab[2] who had worked in the clustering of static images, so some outcomes of the previous work were to be kept in mind so both parts could be joined together.

The GIPSA-lab workflow consists in starting with an idea that is first tested in a simple but complete process and then, when results are evaluated, that process is refined and includes more complexities. The first approach to video motion required clustering a dataset of 20 or 30 shots into three or four differentiated groups.

The clustering algorithm can rely on two types of motion features: camera motion and motion of elements in the scene. Although some work had been done concerning the motion of elements, there were no useful results available, so the requirements for this thesis focused in the analysis of the camera motion.

The original software developed by the Grenoble INP analysed static images using histograms of colour and orientation (texture) and later the different images were grouped using active learning and a technique called Transfer Belief Model[3]. A similar process had to be followed by the rest of features so it could be integrated in a more complete system.

The GIPSA-lab selected a software to extract the camera motion between frames, so the first step of the analysis was already implemented. The used software provides as a result the motion between each pair of analysed frames, but this information cannot be used to calculate the similarity between videos without some post-processing. A histogram of that information is a simple way to represent the information of a shot.

[2] H. Goëau’s work can be read in [2] and [3] or in a short article about it that was published in bitsearch blog, Annex [A].

[3] An good introductory explanation to TBM can be found in [11].

As the software was not developed by the GIPSA-lab, the first step was analysing the performance of the program, evaluate the results and check if they could be used directly or needed any kind of post processing before the histograms were done.

The second step was defining the histogram number of bins and they sizes, so they represented the semantics in the scene as accurately as possible. The bins could be all the same size, present different sizes adapted to the content or even be relative to the frame size.

Finally, the third step of the design was trying different distances to compare the histograms and find the one that provided better results. As the previous work done by H. Goëau was based on the same idea of comparing histograms, there was a list of distances that had been already tried in static image with good results, but they had to be tested as well for videos.

3. Working Plan

The working plan is represented in Fig.1 using a Gantt diagram; the referred tasks are described below:

- Read: Documentation and related work.
- Prepare dataset: Select some scenes for the different purposes or create some synthetic sequences.
- Try software: learn how to use the provided software, the options available…
- Calculate features: Extract motion features and create histograms.
- Preliminary results: Calculate distances between shots.
- Activity in scene: Use difference between images to try to model what happens in the scene.
- Short presentation: Prepare a short presentation for a meeting with all the students of the department.
- Feature extraction: Add the new work to the already existing feature extractor.
- Clustering results: Try K-Means algorithm using different options.
- Write report: Write intermediate report and final report.
- Oral Presentation: Prepare the oral presentation: the slides and choose the information to explain.
- Comment code: Comment the code and write a short manual for following students

Until that point is the part of the project done in Grenoble (week 21). Later in Barcelona the work to be done was running accurate experiments, and it had to be combined with the working activity:

- Revise report: Re-write report to be presented at UPC

- More experiments: Do more experiments in order to complete the report to be presented at UPC.

- Oral Presentation: Prepare an updated presentation for the UPC jury.

Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Read	🟦	🟦	🟦																		
Database		🟩				🟩			🟩					🟩							
Try Software			🟪	🟪	🟪																
Calculate Features					🟦	🟦	🟦	🟦	🟦	🟦											
Preliminary Results								🟩	🟩	🟩	🟩	🟩	🟩								
Activity in scene											🟪	🟪	🟪	🟪	🟪	🟪	🟪				
Short Presentation																🟦	🟦				
Features extractor																	🟩				
Clustering results																		🟪	🟪		
Write Report							🟦	🟦	🟦							🟦	🟦	🟦	🟦	🟦	
Oral Presentation																			🟩	🟩	
Comment code																					🟪
Revise report																					
More experiments																					

Week	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
Read																					
Database																					
Try Software																					
Calculate Features																					
Preliminary Results																					
Activity in scene																					
Short Presentation																					
Features extractor																					
Clustering results																					
Write Report																					
Oral Presentation																	🟩	🟩	🟩	🟩	
Comment code																					
Revise report	🟦	🟦	🟦	🟦	🟦					🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
More experiments						🟩	🟩	🟩	🟩												

Fig.1 Gantt diagram with the working plan

4. State of the Art

4.1 Motion Features

The aim of video analysis is detecting when an event occurs: a person jumping, two people shaking hands, a hit in a tennis match… or sometimes more complex events, such as making a cake or assembling a shelter. Previous works have obtained promising results in action recognition using different approaches.

Interest Points [1, 8, 10, 8]

The most popular features in video analysis consists in finding the *Spatial Interest Points*[4] (SIP) or Spatio-Temporal Interest Points (SITP), or sometimes both. These points are especially interesting because they concentrate information that is spread in the whole image or sequence, so the analysis of these points provides enough information with no need of analysing every single pixel in the image.

In static image analysis there are different techniques to detect the interest points in the scene, SIP, but the most common is Harris Detector, which proposes the use of the following *saliency function*:

$$R(x,y)=det(H(x,y))-k\cdot trace(H(x,y))^2 \tag{1}$$

where *k* should be empirically adjusted (typically between 0.04 and 0.15) and *H(x,y)* is the following matrix:

$$H(x,y)=\begin{pmatrix} \frac{\delta^2 I}{\delta x^2} & \frac{\delta^2 I}{\delta x \delta y} \\ \frac{\delta^2 I}{\delta x \delta y} & \frac{\delta^2 I}{\delta y^2} \end{pmatrix} \tag{2}$$

[4] *Spatial Interest Points* is a technique originally designed to analyse single images and its temporal extension is the one meant for video analysis, as it takes into account the dynamics of the video to find relevant points.

Harris Detector is a corner detector, so it detects those points where the gradient is significant in more than one direction.

Although SIPs can provide good results when working with video sequences, there is a spatio-temporal extension of Harris Detector (STIP) that adds a third dimension, so the points are both relevant in space and time. The detector for STIPs was proposed in [1] in order to find a few points that could be related to the spatio-temporal events of a sequence. The used equations are analogue to the previous ones; the only difference is that the gradient is calculated using also the temporal dimension, so the temporal evolution of the images can be taken into account, too.

$$R(x,y,t)=det(H(x,y,t))-k\cdot trace(H(x,y,t))^3 \tag{3}$$

$$H(x,y,t)=\begin{pmatrix} \frac{\delta^2 I}{\delta x^2} & \frac{\delta^2 I}{\delta x \delta y} & \frac{\delta^2 I}{\delta x \delta t} \\ \frac{\delta^2 I}{\delta x \delta y} & \frac{\delta^2 I}{\delta y^2} & \frac{\delta^2 I}{\delta y \delta t} \\ \frac{\delta^2 I}{\delta x \delta t} & \frac{\delta^2 I}{\delta y \delta t} & \frac{\delta^2 I}{\delta t^2} \end{pmatrix} \tag{4}$$

Although according to [14] STIPs detect much better the *human eye position*[5] in most sequences than SIPs, both SIPs and STIPs can provide relevant information depending on the situation analysed.

Fig.2 is an example of each technique for the frame of a sequence, extracted from [14]. In a sport event the attention is centred in the players, so the people that are moving. In this example STIPs are concentrated in the player, while in the SIP image the attention is also centred in the banner that produces a marked contour.

[5] In [14], SIP and STIP are compared to the human eye position in order to detect the relevant regions of the shot.

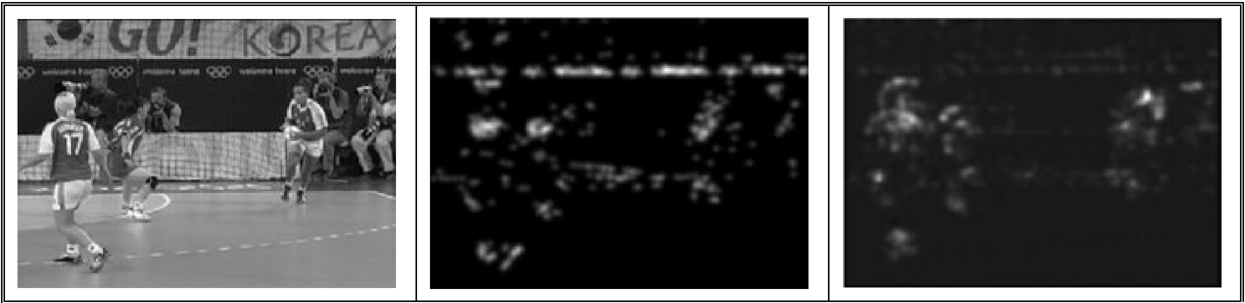


Fig.2 Original image, SIPs and STIPs of the same frame

Bag-of-Features [8, 15]

This technique is an adaptation of the Bag-of-Words (BoW) technique, but applied in image or videos. The BoW is used to characterise a text document (or sentence) as an histogram defined on a vocabulary (or codebook) containing all possible words of interest. For example:

	John reads books, but Matt reads comics.	I never read books.	Distance
Books	1	1	0
but	1	0	1
comics	1	0	1
I	0	1	1
John	1	0	1
Matt	1	0	1
never	0	1	1
read	0	1	1
reads	2	0	2
			Total: 9

Table 1 Bag-of-Words example

Each sentence is characterised by a vector with word counts, so a distance between them can be measured and it is possible to assess “how different” they are. The same idea is used in Bag-of-Features (BoF) technique, but instead of using textual words there is a *visual vocabulary* to work with.

When working with images there are different techniques that find interest points or regions, or other techniques that work with the whole image, but in all those cases the procedure is the same. After being found, those points are characterised using some descriptors. For example, in [15] they use *SIFT descriptor*[6] and in [8] they use the Histogram of oriented Gradient (HoG) and Histogram of optic Flow (HoF). According to the authors of [8], HoG and HoF concatenated are similar in spirit to SIFT, but there are many other descriptors that could be used.

When all regions or points are characterised by a set of features, the next step is creating a visual vocabulary or *codebook* by grouping the most similar features, so each group of features defines a *visual word*. Continuing with the text parallelism, words such as *read*, *reads* or *reading* would be grouped together as they are very similar, so the previous example would be:

	John reads books, but Matt reads comics.	I never read books.	Distance
books	1	1	0
but	1	0	1
comics	1	0	1
I	0	1	1
John	1	0	1
Matt	1	0	1
never	0	1	1
read, reads	2	1	1
			Total: 7

Table 2 Bag-of-Words example

In supervised learning, the codebook is created during the training step and then, when analysing new data, features are assigned to the most similar word in the codebook, no new words are created.

Space-time Shapes [4]

This technique is much less popular than the previous ones, and very specific for human actions. This method uses properties of the solution to the Poisson equation to extract space-time features such as local space-time saliency, shape structure and orientation. According to the authors, these features are useful for action recognition, detection and clustering.

[6] *SIFT* (Scale Invariant Feature Transform): Algorithm to detect and describe local features in images, first published by David Lowe in [1]

The base of this method is the observation that in video sequences a human action generates a space-time shape in the space-time volume. These shapes are induced by a concatenation of 2D silhouettes that contain the spatial information about the position of the body as well as the dynamic information such as global body motion or motion of the limbs with respect to the body. The space-time salient points are detected and then the Poisson equation is used to characterise them by their orientation and other aspects of the space-time shape.

One interesting advantage of that method is that in videos, the extraction of a space-time shape can be simple in some situations such as surveillance videos, where with a difference algorithm it is enough to extract satisfactory space-time shapes. Using this technique the usual segmentation problem can be avoided.

Multiple Views [6]

Although this system requires an event recorded by more than one camera, most sport competitions are recorded by several cameras, so it could be very useful in the analysis of this type of sequences.

Having more than one point of view makes it possible a completely new approach. Given a type of low level features, the distance between extracted features is computed for each pair of frames and the results are stored in a Self-Similarity Matrix (SSM).

For a sequence of images I={I_1, I_2, …, I_T} in discrete *(x,y,t)*-space, a SMM of *I* is a square symmetric matrix of size *T*x*T* as follows:

$$[dij]_{i,j=1,2,\ldots T} = \begin{pmatrix} 0 & d_{12} & d_{13} & \cdots & d_{1T} \\ d_{21} & 0 & d_{23} & \cdots & d_{2T} \\ \vdots & \vdots & \vdots & & \vdots \\ d_{T1} & d_{T2} & d_{T3} & \cdots & 0 \end{pmatrix} \qquad (5)$$

where d_{ij} is the distance between the low level features extracted in frames I_i and I_j respectively.

In the article the SSMs are represented graphically instead of only numbers, and it is observable some similarities when the matrices correspond to the same event. The patterns of the proposed SSMs are quite stable through changes of viewpoints, and this can be used for posterior action recognition.

4.2 Distance measuring techniques

The distances previously tested by H. Goëau [2] for the static images analysis (colour and orientation) were the Manhattan, Euclidean and Bhattacharrya ones. In this section presents them, as well as other two distances also popular in video retrieval.

Manhattan distance (L1)

Absolute distance between each component of a vector.

$$dL_1(h1, h2) = \sum_{i=1}^{n} |h_1(i) - h_2(i)| \qquad (6)$$

Euclidean distance (L2)

“Ordinary” distance between two points, it is given by the Pythagorean formula ($h^2=c_1{}^2+c_2{}^2$).

$$dL_2(h1, h2) = \sqrt{\sum_{i=1}^{n} (h_1(i) - h_2(i))^2} \qquad (7)$$

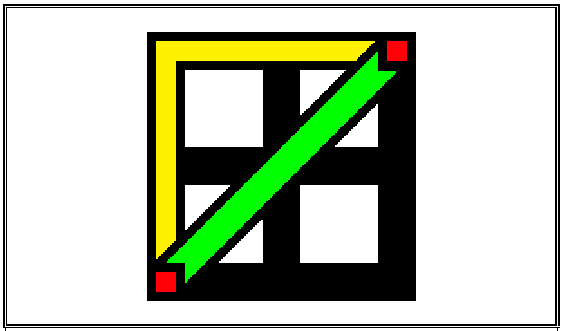


Fig.3 Euclidean (green) and Manhattan distance (yellow) between two points.

Bhattacharyya distance

Used when comparing two discrete probability distributions, commonly used in computer vision.

$$d_{Bhat}(h_{1,} h_2) = -\log \sum_{i=1}^{n} \sqrt{\frac{h_1(i) \cdot h_2(i)}{|h_1| \cdot |h_2|}} \quad (8)$$

Cosine similarity [15]

Two vectors can be compared measuring the cosine of the angle between them, depending on the result it is possible to know whether they are pointing to a similar direction or they are completely independent. This similarity measure is often used in text retrieval, but could be an interesting option for the kind of data used in this project.

The cosine similarity is calculated using the Euclidean dot product:

$$a \bullet b = ||a|| ||b|| \cos \theta \quad (9)$$

It is possible to isolate the cosine that will have a value between 1 and -1. Meaning 1 exactly the same, -1 exactly opposite and 0 independence. In cases where vectors have no negative values, the value of the cosine similarity is restricted to [0,1].

Earth mover's distance (Wasserstein metric in Mathematics**)**[13]

In computer science the Earth mover's distance is used to measure the distance between two probability distributions.

$$EMD(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij}} \quad (10)$$

The Earths mover's distance is informally explained as two piles of "dirt" over a region D, the EMD is the minimum cost of turning one pile into the other. This cost is the distance to be moved multiplied by the amount of "dirt".

4.3 Video Domains

There exist several different types of video domains, each of defining certain particularities. Some of them have raised special interest between the scientific community:

- News bulletins: Most news programs present two basic types of shots: the shots where there is the TV anchor, only head and shoulders; and then the clips showing the news. The first type of shots can be detected quite easily, so finding the beginning and the end of the different clips can be done automatically.

- Movies: In movies, the images are planned and well illuminated and the actions are clearly presented, so analysing the content of those images is easier than in other situations.

- Sports: This kind of events are usually recorded by more than one camera, so there is the same action from different points of view, making possible new types of analyses that would not be possible in the other type of content. Furthermore, in some sports there are some characteristic sounds that can bring a lot of information of the actions performed. There is an example of the multiple view technique in [6].

5. Design

The original idea for that project was adding a video analysis module to the system that already existed.

Fig.4 shows the block diagram of the proposed system at the beginning of the internship. Given a video shot, its camera motion had to be extracted for two purposes:

- Create the motion histograms that would be later used in the TBM module
- Compensate the camera motion in order to compare consecutive images and detect the elements in motion in the scene.

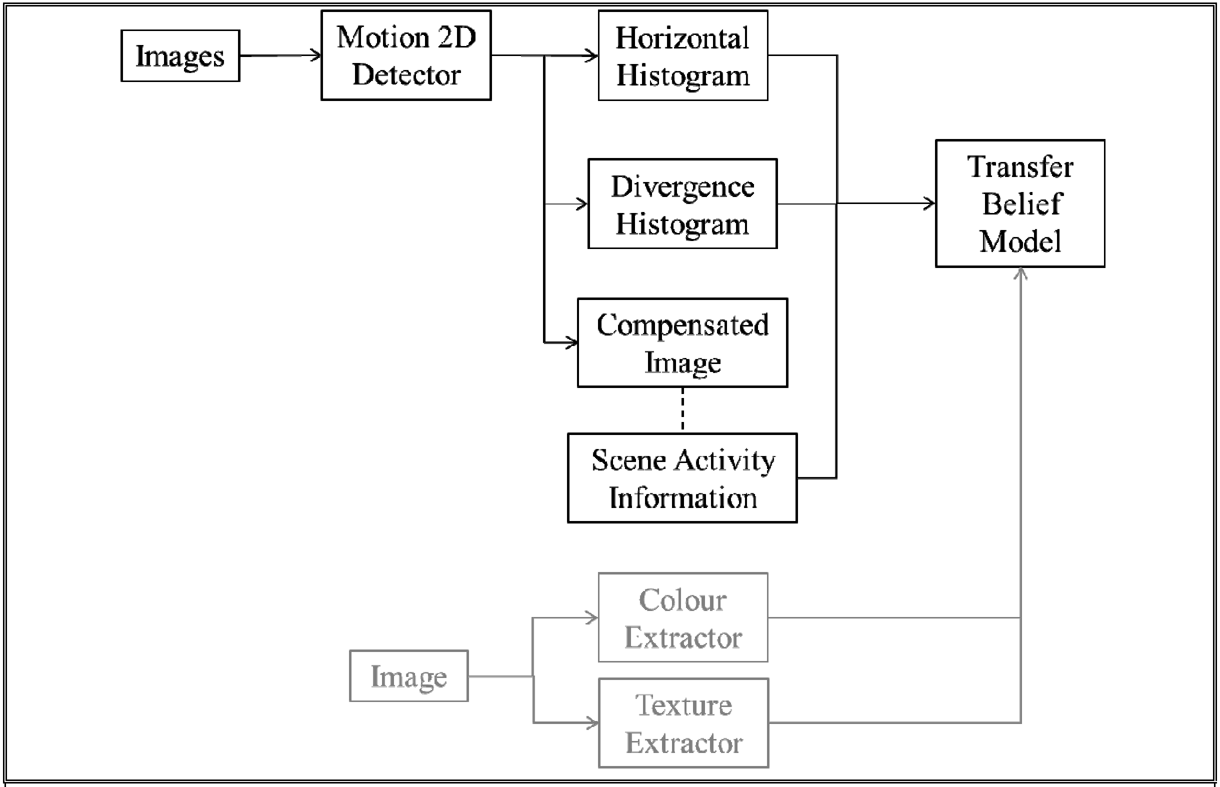


Fig.4 Representation of the original idea

Due to a lack of time, the original design was simplified by the one in Figure 5. The system depicted, only analyses the camera motion and runs some tests to show that clustering is quite successful using only that information. Future work could add the camera motion module to the complete system and using compensated images to extract some information concerning the scene activity.

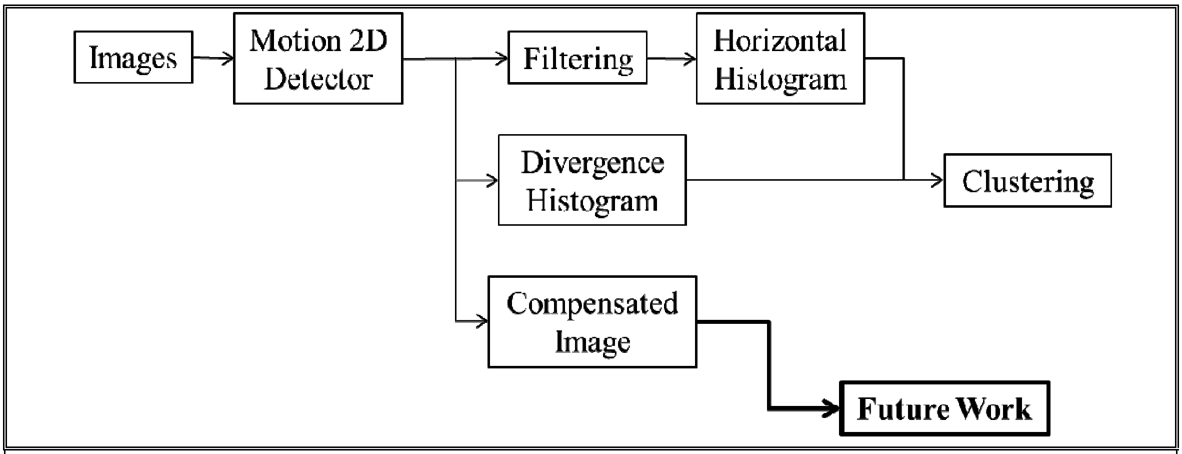


Fig. 5 Representation of the actual design

5.1 Features selection from Motion2D extractor

The GIPSA-lab has chosen a camera motion extractor, Motion2D, which is the basic tool to obtain the features to work with. Developed by Vista team of INRIA[7], this previously existing software analyses a pair of frames and models the dominant motion occurred between them assuming the illumination in the sequence is constant, or almost (as the program itself can detect the change of illumination between two frames). It generates a robust multi-scale estimation from the spatio-temporal gradients of the luminance of the image.

There are different models available: constant, affine or quadratic, and some simplifications to avoid unnecessary calculations. The constant model only uses the motion parameters c_1 and c_2 and the quadratic model has 12 parameters to show transformations that cannot be performed with a regular camera, so in this work the model chosen is the affine model with six parameters:

$$\vec{\omega}_A(p_i)=\begin{bmatrix} u(p_i) \\ v(p_i) \end{bmatrix}=\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}+\begin{pmatrix} a_1 a_2 \\ a_3 a_4 \end{pmatrix}\cdot\begin{pmatrix} x_i \\ y_i \end{pmatrix} \qquad (11)$$

[7] INRIA: Institut National de Recherche en Informatique et en Automatique. French national research institution focused on computer science and applied mathematics.

where $p_i=(x_i,y_i)$ is the pixel position, c_1 and c_2 are the *horizontal* and *vertical translation* respectively, and the *a* parameters are used to calculate *divergence* and *rotation*.

Parameters *c1* and *c2* are the number of pixels the scene has moved, but the other two types of motion are not extracted directly.

Divergence, which indicates if the scene is getting closer or further, is calculated as:

$$\mathrm{div}=\frac{1}{2}(a_1+a_4) \tag{12}$$

Rotation is calculated as follows, returning the sinus of the rotation angle used:

$$\mathrm{rot}=\frac{1}{2}(a_2-a_3) \tag{13}$$

As simplicity is one of the requirements for the system, only two of the four possible motions are analysed: *horizontal translation* and *divergence*. *Rotation* was discarded because it is not common, only in subaquatic documentaries or in some special camera effects in a film, so it is not studied at all. About *vertical translation*, it is more popular than rotation, but it does not bring much semantic information, as an element moving vertically is not usually followed by a camera. Due to Human Visual System field of vision (15:9 ratio approximately), cinema and television screens are wider than tall making vertical tracking much difficult for the spectator. In most cases when an element goes through a scene the camera remains still while the element crosses the screen, too. When a camera is been moved vertically, very often it is only to show a scenario, like a horizontal travelling to show a mountain range but vertically and to show a cliff, for example.

Upwards travelling	**Downwards travelling**
Initial frame	Initial frame
Final frame	Final frame

Fig. 6 Two examples of vertical travelling in films.

Due to all this and after a few tries, it was considered that most of *vertical* motion was caused by hand shaking when the camera is handheld, as it happens in Fig. 7.

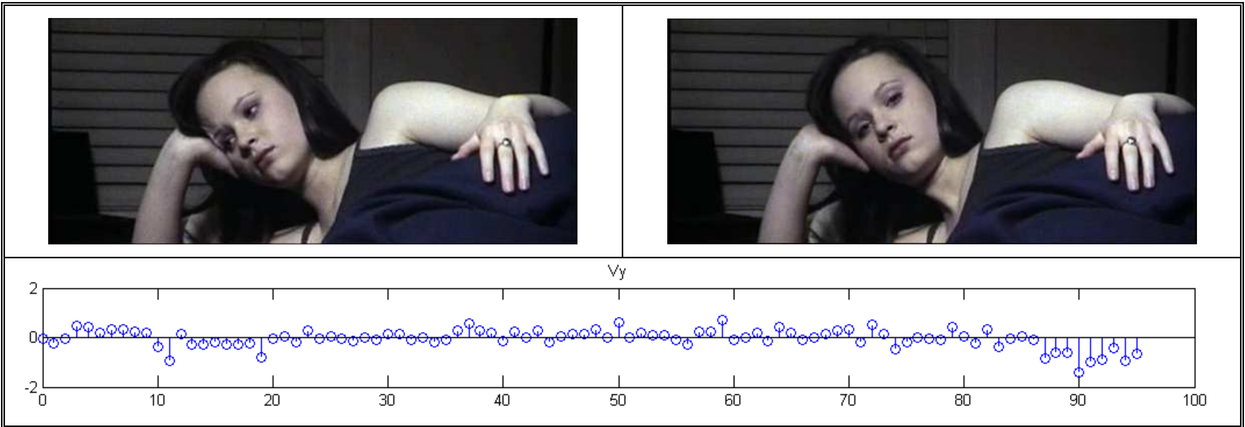


Fig. 7 Random frames of a handheld shot and the vertical motion detected due to shaking.

Combining both horizontal translation and divergence it is possible to model different types of camera motion techniques:

- Pan: The camera turns horizontally keeping its position to follow an element moving fast or show a complete scene. It is detected as horizontal motion and also a small divergence.

- Zoom: Technically it is not a camera move, but a change in the lens’ focal length gives the illusion of moving the camera closer or further of the scene. It is detected as divergence if the zoom is centred or it can also include some horizontal translation if it focuses on a non-centred part of the scene.

- Dolly: The camera is mounted on a cart which travels along tracks. Also known as a tracking or trucking it is detected as horizontal motion if it sweeps the scene, as divergence if the camera moves straight to the centre of the scene, or as horizontal motion and divergence if the camera gets closer to an element that originally was on one side.

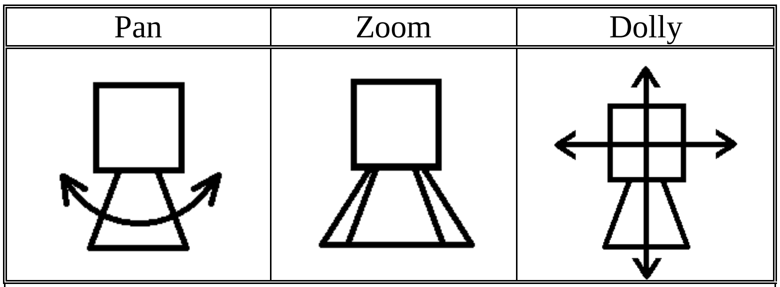


Fig. 8 Types of camera motion

5.2 Horizontal motion

Horizontal motion can provide valuable information related to the semantics in the scene. The first two images in Fig. 9 are extracted from sequences where the character is running and the camera is moving so fast that even the image is a little blurry. As in both sequences the camera follows men running, the camera motion is much faster than in the third example, which corresponds to a slower sequence where the man is entering the house walking.

Fig. 9 Images extracted from Forrest Gump, The Graduated and The Pianist, respectively.

In Fig.10, there is a schema of the different operations performed to the horizontal Motion2D results. For all images of the sequence the $|c_1|$ parameters are concatenated in a vector and then filtered before the histogram is built.

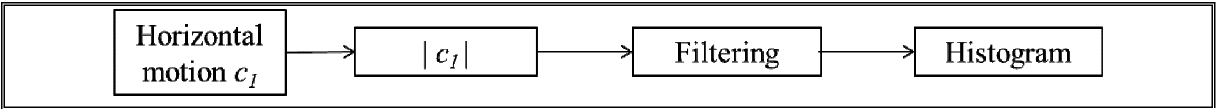


Fig.10 Horizontal motion processing

Absolute Value

For a human observer, intuitively, the direction of a horizontal motion does not contribute to the semantic interpretation of the action in a sequence; the relevance relies on the speed of that motion. Two sequences with a camera moving fast, either to the right or to the left, seem much more similar between them than a slow travelling. Fig. 9 shows some examples of popular films to illustrate this. From a semantic perspective, the first two sequences where the men are running would be considered more similar than the second and the third, where two men go to the left. That is the reason why in horizontal motion analysis only the absolute value of c_1 is used.

Filtering the signal

In some sequences some peaks and valleys appear that do not correspond exactly to the real camera motion and that must be deleted. In Fig.10 and Fig.12 there is an example of a peak and the situation that cause it.

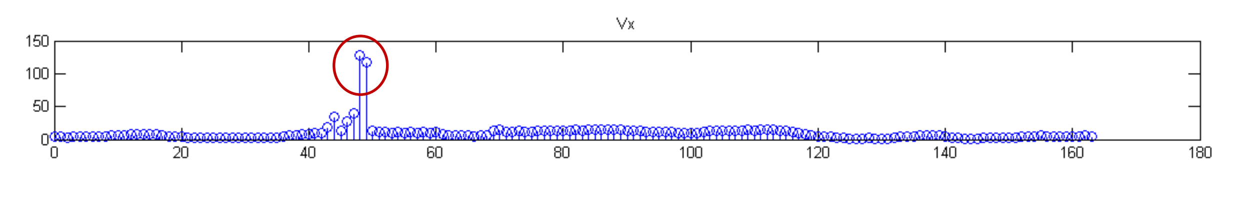


Fig.11 Horizontal motion of the previous clip: Detected motion (*Vx*).

The sequence is a travelling in a forest, so there is a moment when there is a big tree in foreground covering almost all the scene. The problem caused by big objects is that if they occupy more than 50% of the image, its pixels are the ones used to estimate the dominant motion.

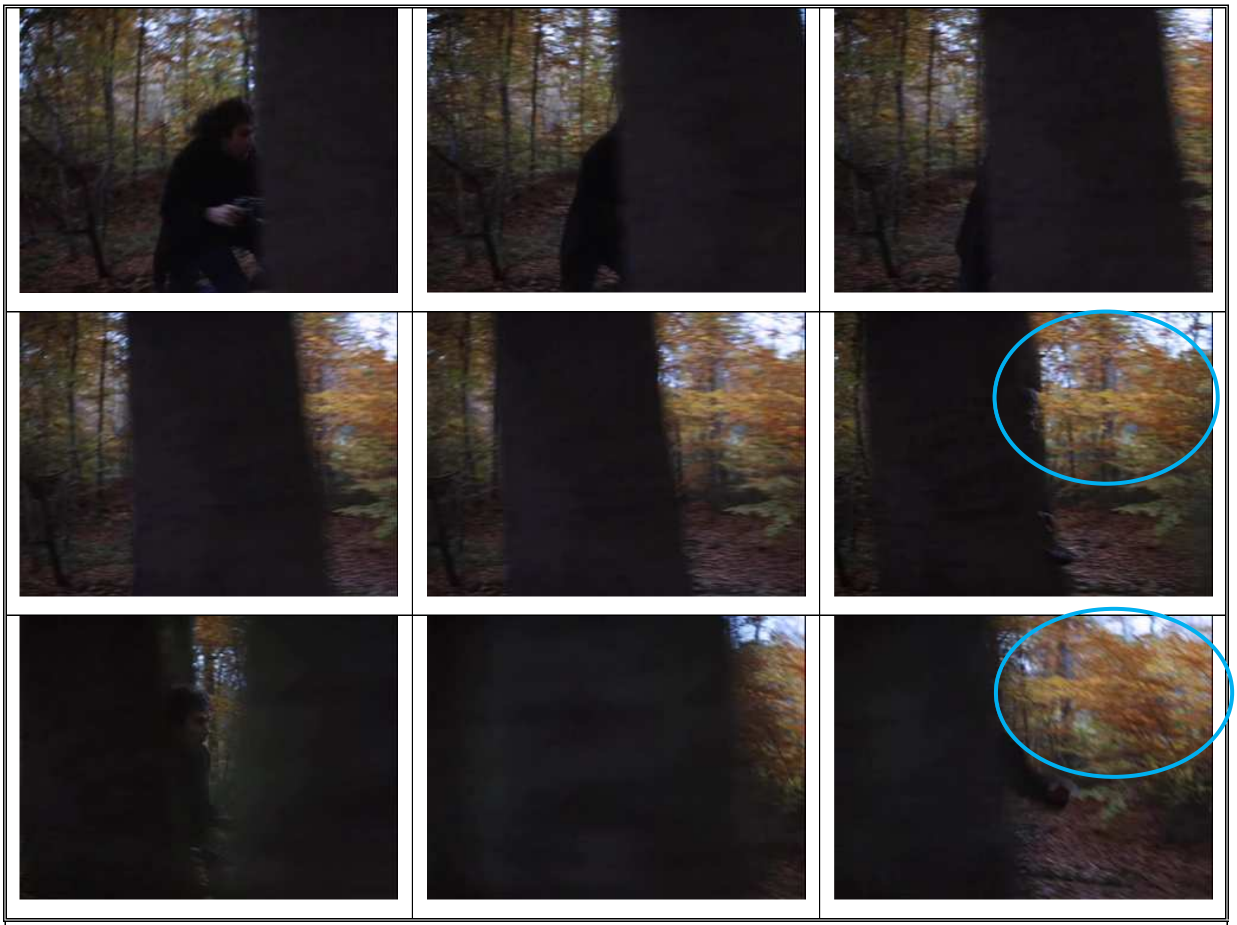


Fig.12 Frames 42 to 50 of a horizontal travelling.

It is observable that before and after the large tree, the plant circled in blue in Fig.12 is the same, so the real camera motion has not been as high as it has been calculated; this increase of speed is only due to the occlusion of the scene because of the tree.

An object in foreground can also create the same problem but in the opposite sense, instead of having peaks, valleys. Fig.13 is an example of that problem:

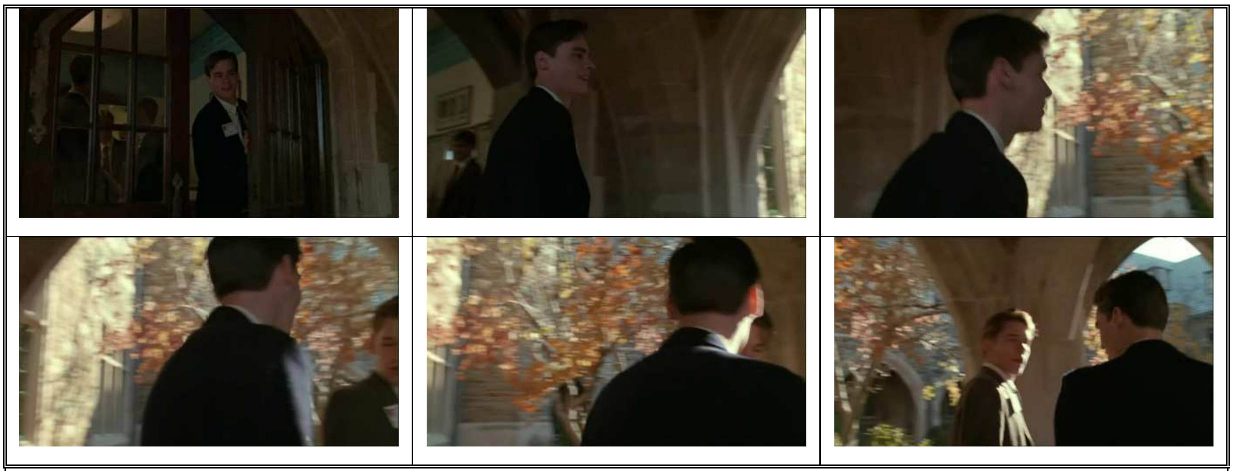


Fig.13 Frames 0, 30, 43, 52, 57 and 70 of the sequence

In this sequence the camera follows the person that crosses the scene, but instead of a sliding camera, it turns horizontally, this creates problems around frame 50 when the person is closest to the camera. In this case the foreground object is detected in the opposite direction of the real motion.

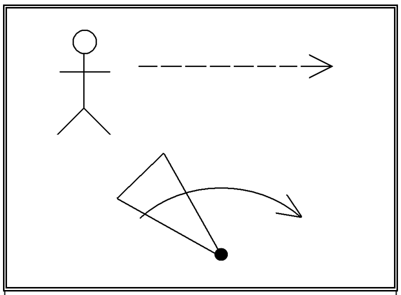


Fig.14 Schema of the problematic camera motion

It can be observed that in the middle frames of Fig.12 the person appears much closer to the camera than the background, or in some cases the image is not even well defined. This situation creates the sequence shown in Fig. 15.

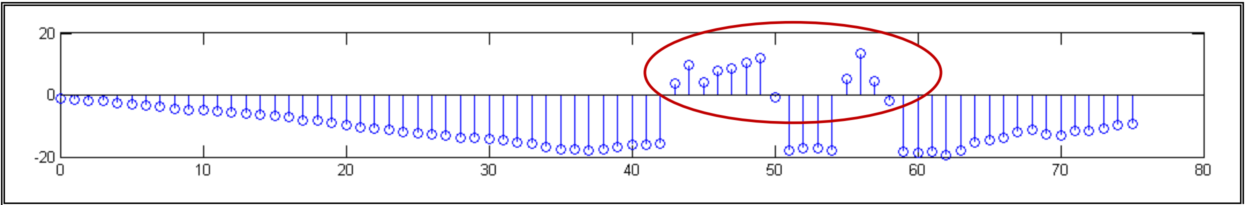


Fig. 15 Motion graph, real value[8]

Although this sequence is a right travelling, there are some moments where it seems the camera moves to the left, which is not true in this scene. This confusion is due to a change of the pixels used to estimate the motion, as it happens in the tree scene. The absolute value of the motion presents some valleys that have to be filled to obtain the real camera motion.

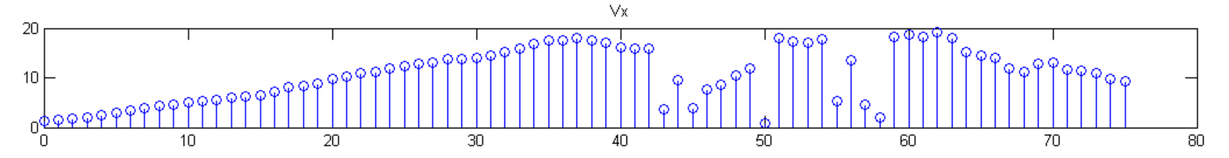


Fig.16 Motion graph, absolute value

The scene filtering softens and even deletes the peaks and valleys, depending on the type of filter used. The choice of the best filter is also conditioned by the following stage: the computation of a motion histogram.

Fig.17 contains the graphs of the calculated motion and two different filters to try to eliminate the peak caused by the tree. According to [5], 13 frames correspond to 0.5 second, which matches the human reflex time. This figure was chosen to help the system behave similarly to what a real person would do.

[8] In this graph the sign of the motion has been kept to show the real situation.

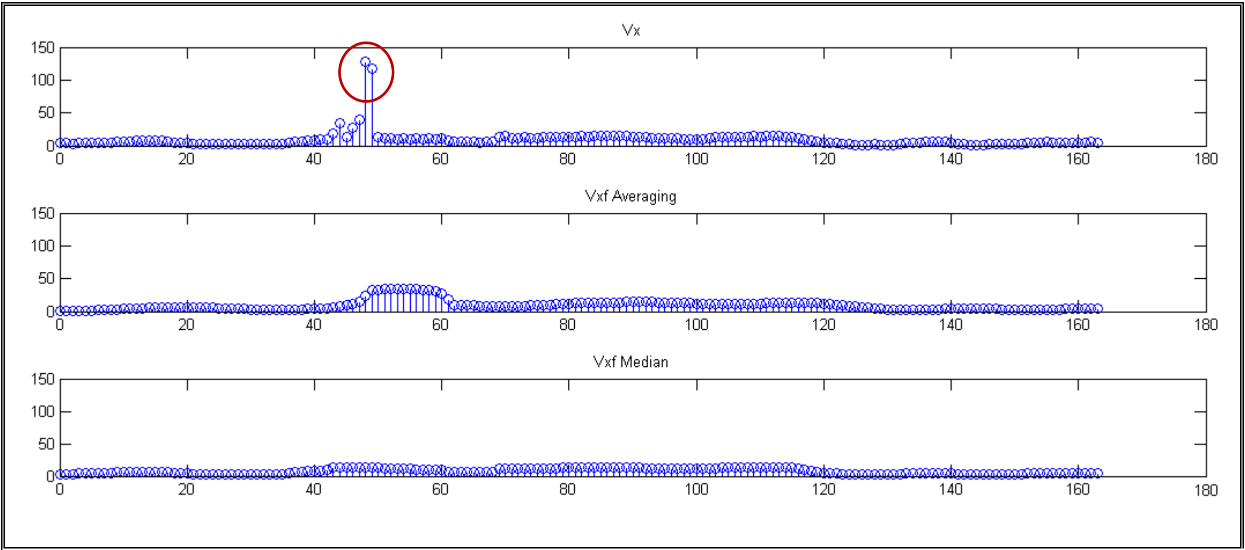


Fig.17 Horizontal motion of the previous clip: Detected motion (*Vx*), average filtered motion (*Vxf Averaging*) and median filtered motion (*Vxf Median*).

In both *Vxf Averaging* and *Vxf Median* the peak is eliminated, but to decide which one of both is the best option, it is necessary to have a look at the resulting histograms study which one performs better for the following operations.

Histogram generation

In order to work with the horizontal motion, it is necessary to work with features independent from the video duration. As the clips may not present exact same length, a frame by frame comparison would be incomplete and, most of times, useless.

A simple way of representing the motion of the camera is a histogram with different bins to represent the different speeds estimated by Motion2D. Normalised histograms, representing the percentage of each bin in the whole sequence, are the proposed solution to compare the video clips independently of their duration by using the distances explained in chapter 4.2.

In the first steps of that project 5 bins were created adapted to the different types of horizontal travelling that were found in Hollywood database. In Fig.18 some commented examples generated from this dataset.

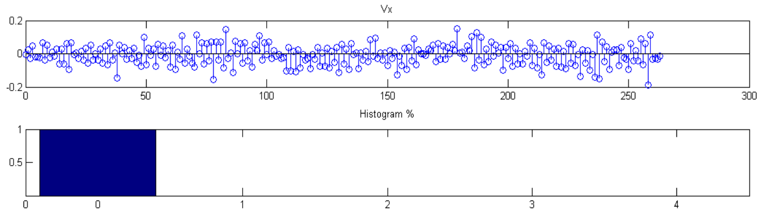
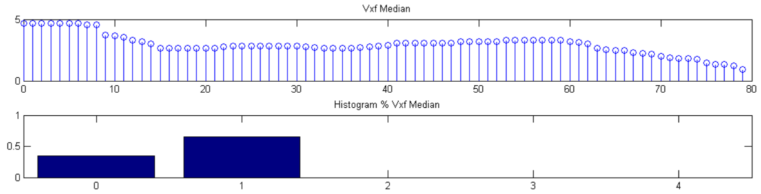
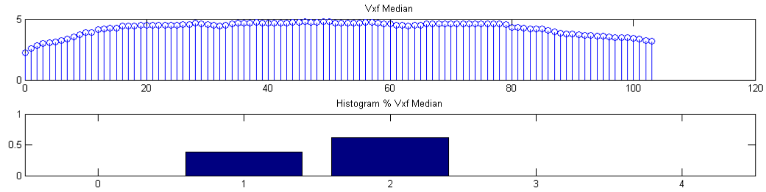
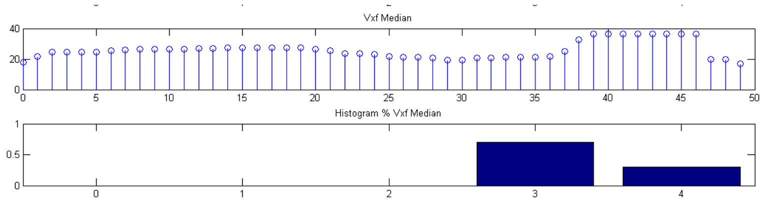
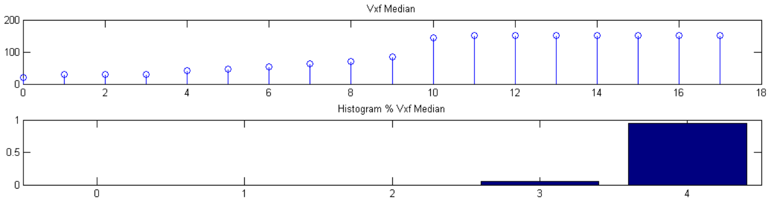
Clip	Motion and Histogram
 Static, the camera is fixed in a position and there is no motion.	
 Slow travelling, to show the scene detailed in contrast with all the activity.	
 Medium travelling, showing a scene with no action	
 Fast travelling, showing a person running	
 Very fast, change of character without changing camera. (fastest shot in all considered datasets)	

Fig.18 Examples of first histograms

Motion2D provides the number of pixels of displacement, so this has to be adapted to create the histogram bins. The bins were not defined based on the absolute values provided by the motion estimator; their limits depend of the relation between those values and the size of the image. This is due to the fact that as bigger is the frame; the more number of pixels the image changes, even though the camera has moved in the same speed. Table 3 includes an initial mapping between the bins and the relative horizontal motion.

Bin number	Size (relative to horizontal size)	Speed
0	0% – 0.5%	Static
1	0.5% – 1%	Slow
2	1% – 2%	Medium
3	2% – 5%	Fast
4	5% – ∞	Very fast

Table 3 Horizontal motion histograms of 5 bins

Although results using 5 bins were quite good[9], further experiments split the first bin in two or three, so slow motion could be better modelled, obtaining the bins defined in Table 4 and Table 5:

Bin number	Size (relative to horizontal size)	Speed
0	0% – 0.25%	Static
1	0.25% – 0.5%	Very slow
2	0.5% – 1%	Slow
3	1% – 2%	Medium
4	2% – 5%	Fast
5	5% – ∞	Very fast

Table 4 Horizontal motion histograms of 6 bins

Bin number	Size (relative to horizontal size)	Speed
0	0% – 0.1%	Static
1	0.1% – 0.25%	Almost static
2	0.25% – 0.5%	Very slow
3	0.5% – 1%	Slow
4	1% – 2%	Medium
5	2% – 5%	Fast
6	5% – ∞	Very fast

Table 5 Horizontal motion histograms of 7 bins

[9] Some 5-bins results can be found in Annex [B]

The results of the two filters previously explained can be evaluated with the histogram bins defined. The graph in Fig.19 shows the original Motion2D data of the tree problem with values in almost all bins, which is not desirable because it does not match the semantic interpretation of the camera motion, as this shot is in continuous motion and the first bins (slow) are not empty. The most important problem to solve is the peak around frame 48, which causes the 6th bin to be not null.

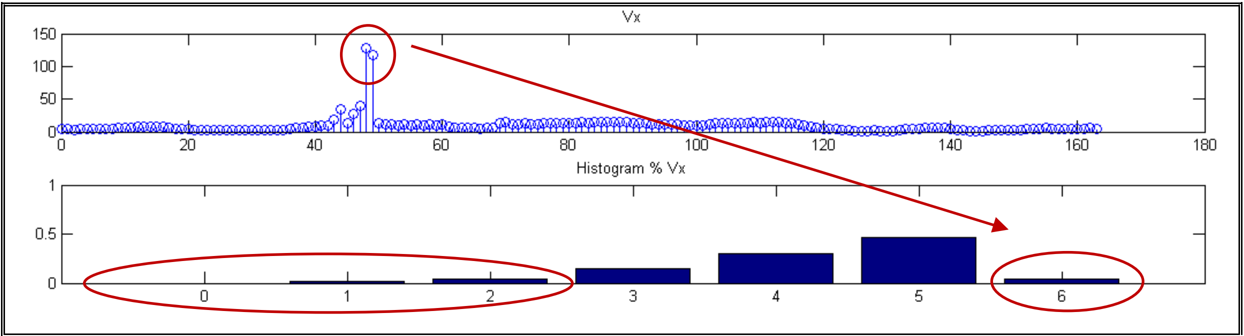


Fig.19 Horizontal motion of the tree clip: Detected motion and its histogram (7 bins)

The averaging filter (Fig.20) makes the slow bins to spread, so the static one now has some value, too and also increases the 6th bin (very fast), which makes the histogram worse than the original one.

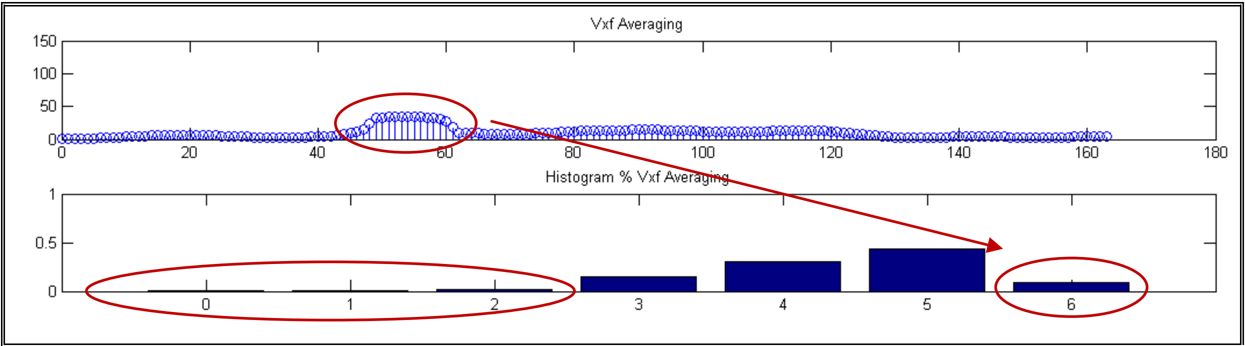


Fig.20 Horizontal motion of the tree clip: Average filtered motion and its histogram (7 bins)

The application of the median filtering (Fig.21) eliminates the peak completely and strengthens bin number 5. The last bin is eliminated, so the tree effect has disappeared, and the slow bins are null as well. The resulting histogram matches much better the expected result.

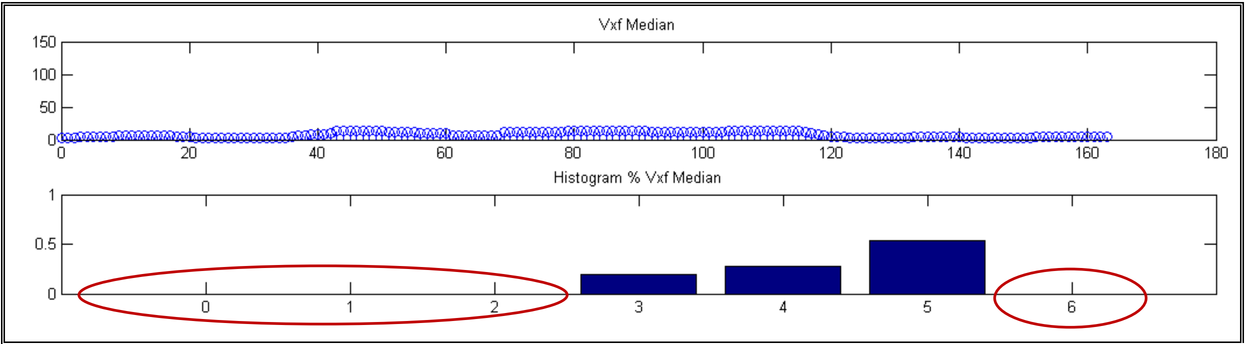


Fig.21 Horizontal motion of the tree clip: Median filtered motion and its histogram (7 bins)

The second example with the boy in foreground is also solved by the median filtering. As it is shown in Fig.22, the filtered sequence does not depict exactly the real camera motion but it is improved enough to obtain a satisfactory histogram.

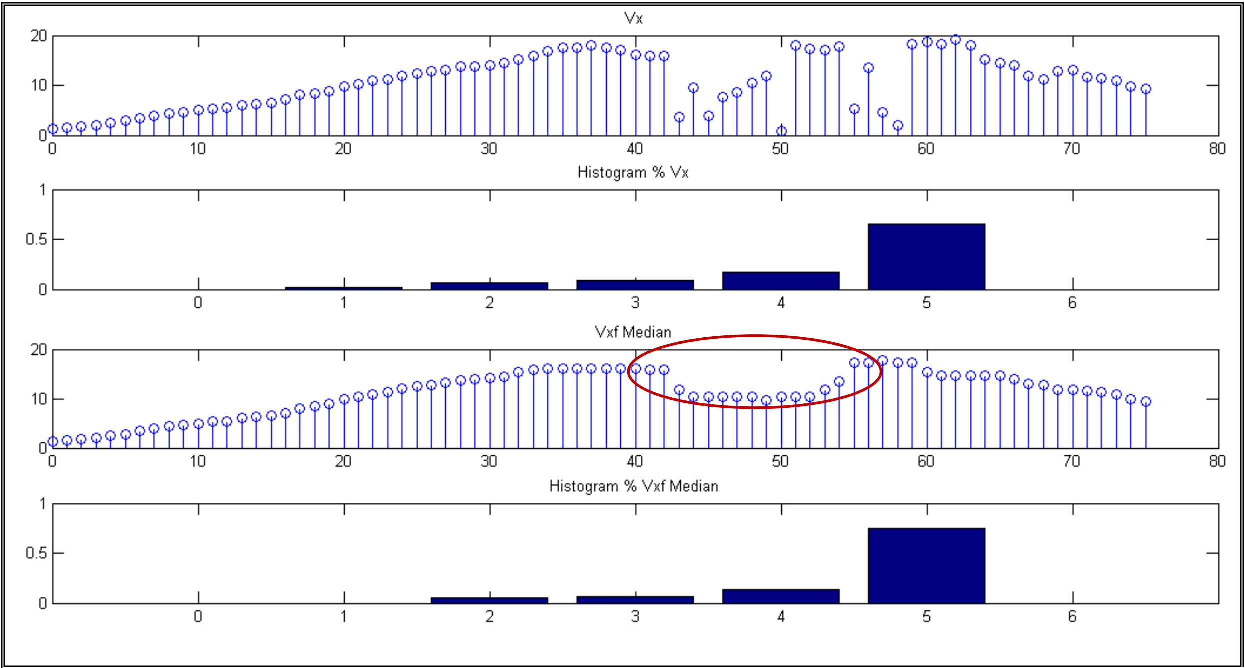


Fig.22 Original motion (*Vx*) and median filtered motion (*Vxf Median*), both absolute, value and their histograms (7 bins)

Applying a median filter provides better data to work with, so the rest of the work presented in this document concerning horizontal travelling presents filtered data.

5.3 **Divergence**

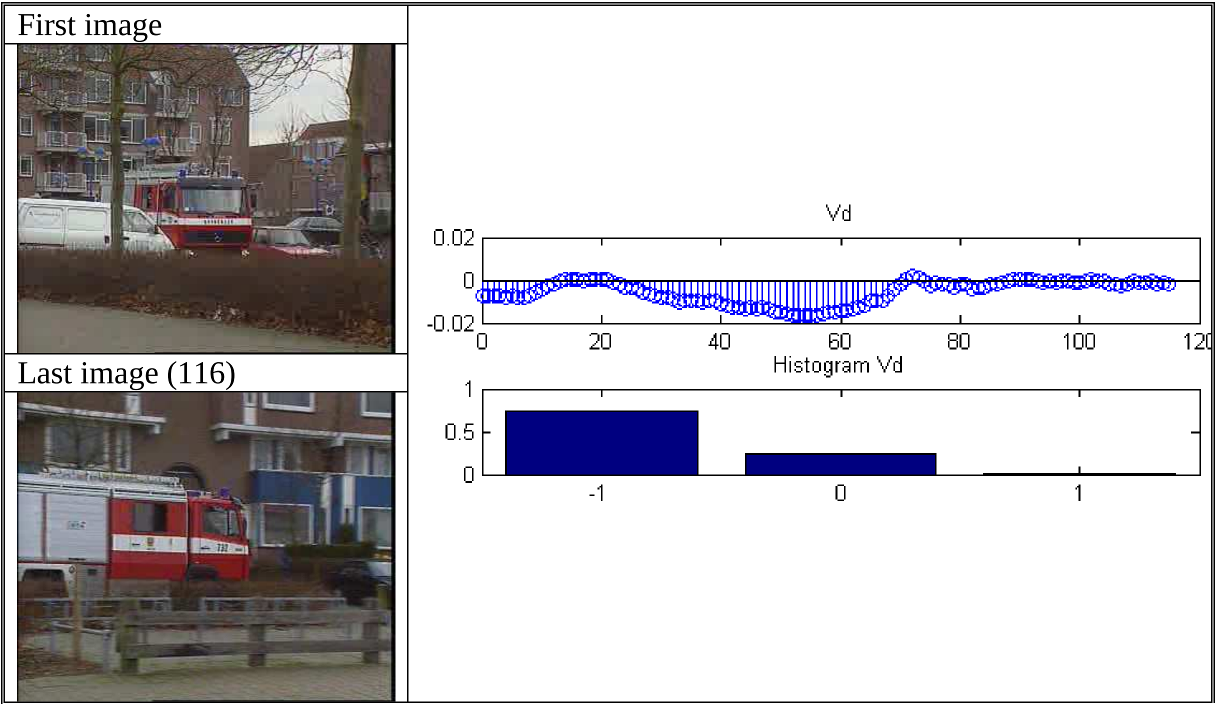
The Proposed system analyses the divergence, in addition to the horizontal motion. The combination of both features can help the system to interpret the video semantics. The divergence of two frames is calculated using the equation 12 and then it is treated similarly to the horizontal travelling but in a much simpler way.

Divergence estimates if in the analysed clip there is a zoom and/or a camera travelling in the z direction; both operations change the spatial scale of the scene, so they are detected together.

The first tests used only three bins, as shown in Table 6:

Bin number	Values	Percentage	Meaning
-1	[-∞, -0.001]	[-∞, -0.1%]	Zoom out or camera going backwards
0	[-0.001, 0.001]	[-0.1%, 0.1%]	No appreciable change
1	[0.001, ∞]	[0.1%, ∞]	Zoom in or camera getting closer to the scene

Table 6 Divergence 3 bins


First image

Last image (116)

Fig.23 First picture and last picture of a shot and its divergence graph and histogram.

In the scene shown in Fig.23, the camera goes to the left but is also makes a zoom out to follow the truck more easily; it is a usual technique in situations as the one explained in Fig.14. It is also usual to make a zoom in again after the person or object being tracked has reached the middle point, but it does not happen in this particular shot. The divergence data did not require any previous filtering, so the histogram is made by using the data provided by Motion2D.

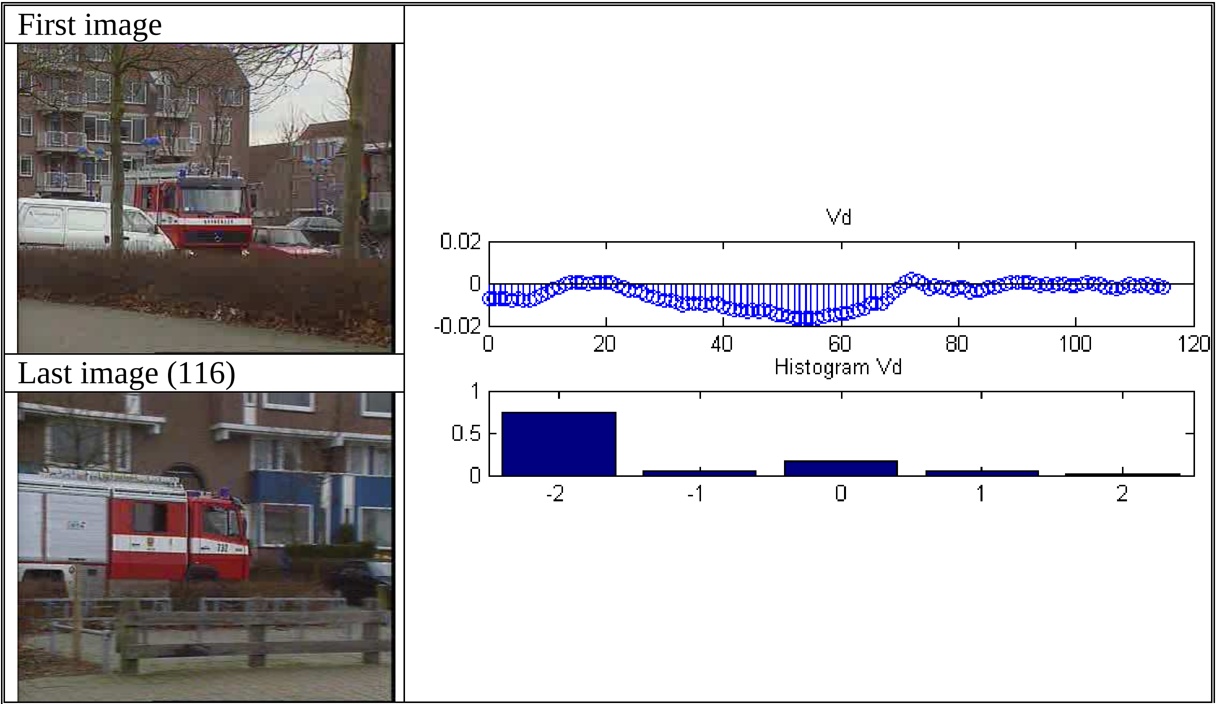
Analogously to the horizontal traveling analysis, in order to improve the results the two no-null bins were split in two[10].

Bin number	Values	Percentage	Meaning
-2	[-∞, -0.001]	[-∞, -0.1%]	Zoom out or camera going backwards fast
-1	[-0.001, 0.0005]	[-0.1%, 0.05%]	Zoom out or camera going backwards slowly
0	[-0.0005, 0.0005]	[-0.05%, 0.05%]	No appreciable change
1	[0.0005, 0.001]	[0.05%, 0.1%]	Zoom in or camera getting closer to the scene slowly
2	[0.001, ∞]	[0.1%, ∞]	Zoom in or camera getting closer to the scene fast

Table 7 Divergence 5 bins

Using the new bins the same sequence:

[10] Some 3-bins results can be found in Annex [B]



First image

Last image (116)

Fig.24 First picture and last picture of a shot and its divergence graph and histogram.

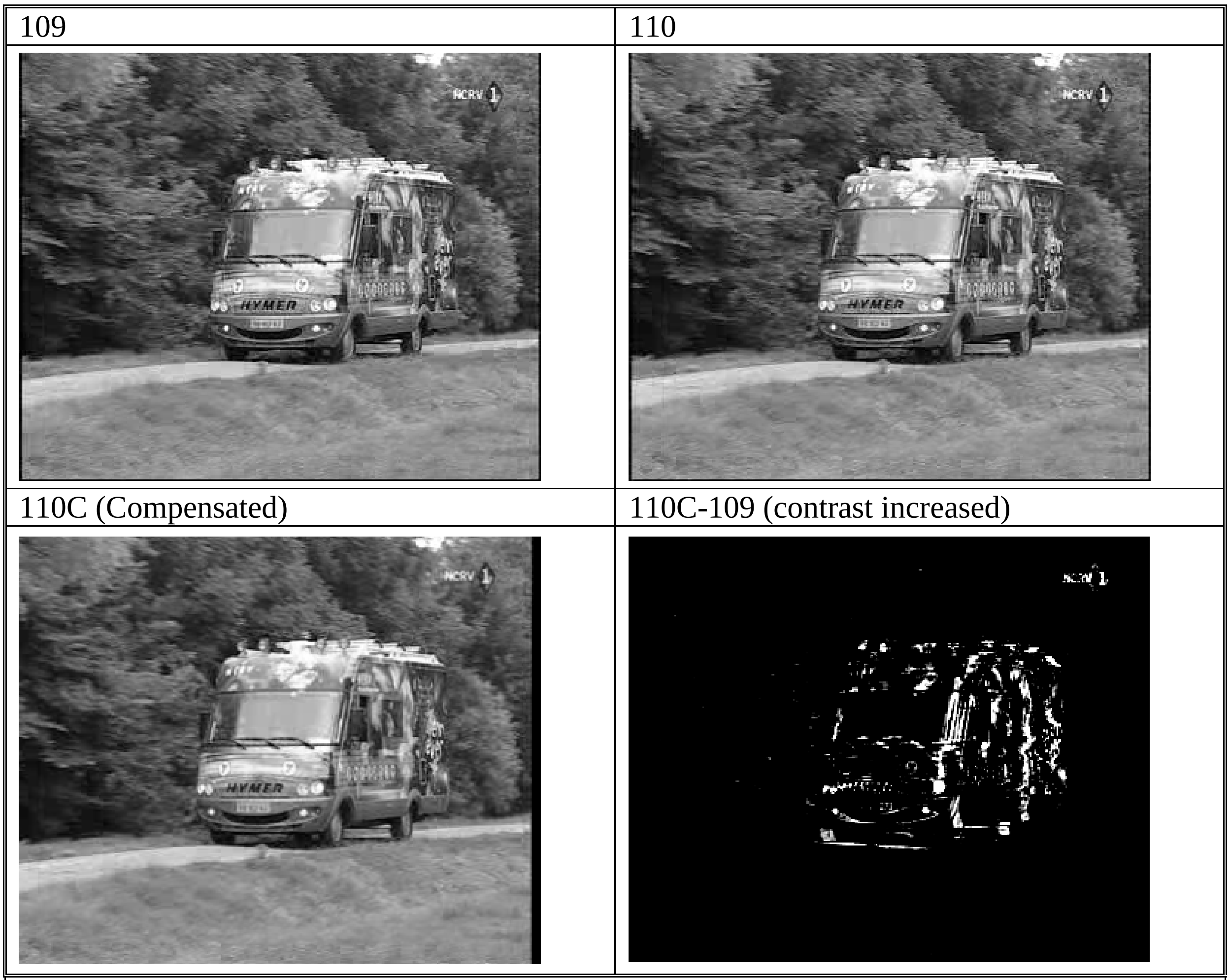
5.4 Rejected designs

During the project there was an attempt of easily model the motion of the elements in the scene. Although no successful results were achieved, it was decided to include this part of the project to show that Motion2D results can be used for other purposes In addition to modelling the camera motion.

Compensated image

The information provided by Motion2D is used to build the motion histograms, but it is also used to generate the compensated images that can be used to detect the motion of the elements in the scene. In a previous work by a former student at GIPSA-Lab, there was an algorithm to compensate the camera motion of two consecutive frames, modifying frame t+1 to be as similar as possible to frame t.

In Fig.25 there is an example of a compensated image and what can be obtained by the difference between image t and t+1 compensated.



109

110

110C (Compensated)

110C-109 (contrast increased)

Fig.25 Compensation and difference of two consecutive frames

In the difference image (110C-109) it can be observed two different light regions:

- Moving element(s): In this example the only element in motion is the van, and it is the element whose motion has to be studied. In case there are more than one object in motion or the objects are deformable (like a person walking), some additional operations would be needed to obtain the motion of each of those elements.

- Synthetically added objects: Any object added a posteriori would fit in this category, like subtitles or the scoreboard of a football match, in the example image it is possible to see the TV station logo in the top right corner. It is not a moving object, in fact it is not even in the scene, so it has to be identified and ignored as it can create "ghost activity" that is not real.

The compensation and difference of the frames can be used to model the motion of the elements in the scene. Two concepts were tested to obtain information of the activity in the scene: Evolution of Barycentre and Evolution of Saliency. Although there were no satisfactory results, some conclusions can be inferred from them.

Evolution of Barycentre

There was an attempt to model the motion of the elements in the scene using the evolution of the barycentre, but results were not good at all.

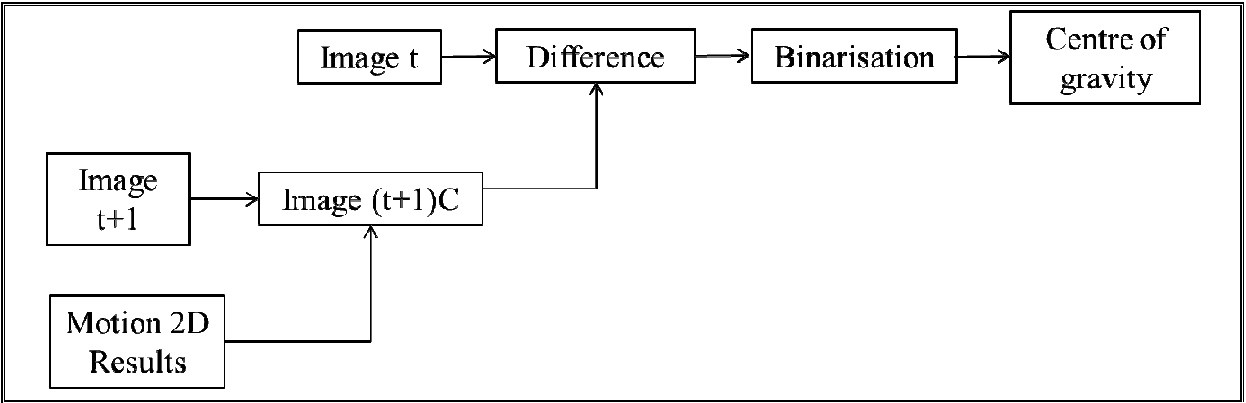


Fig.26 Block diagram of operations needed for barycentre evolution

The first attempt was using the binary difference image; calculate the centre of gravity and then its horizontal evolution along the frames of the sequence. This system seemed to work with synthetic images, as the one considered in Fig.27. In this sequence the girl was moved 10 pixels to the right each time, so the barycentre technique seemed to work.

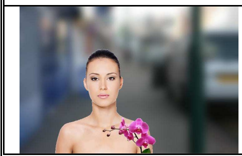
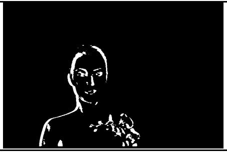
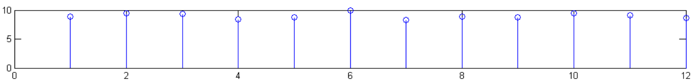
Original image	Binarised difference image	Evolution of centre or gravity
		

Fig.27 Horizontal evolution of centre of gravity in synthetic sequence using difference image

The problem appeared when using real sequences, there was noise and the elements were changing, so results were much more different. Results in real sequences were very noisy and their median value was around zero for all videos used.

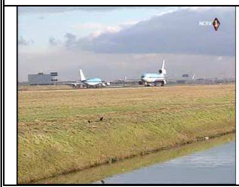
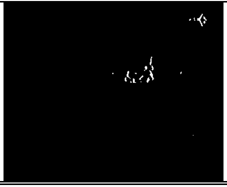
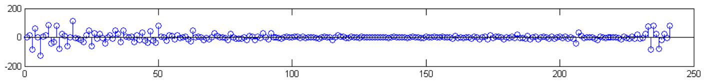
Original image	Binarised difference image	Evolution of centre or gravity
		

Fig.28 Horizontal evolution of centre of gravity in real sequence using difference image

Real sequence results were much worse than the synthetic sequence, because in the synthetic image the moving element did not modify its shape. As the element in motion was the same picture displaced, it remained still and so the white pixels resulting from the difference were exactly the same.

In real sequences the element may change its shape, or simply the light can make the element seem different, so the difference pixels used to calculate the barycentre are never the same, so the comparison is much more difficult.

Evolution of Saliency

To try to obtain better results, a more sophisticated technique was tried. In a previous work [3], a dynamic saliency detector was used to find which areas the human brain finds interesting in a video sequence. The analysis of the motion is done trying to emulate the human visual system following these steps:

1- Compensation of dominant motion

2- “Retinian” filtering. The visual information is treated as it is in the human eye, it is a band-pass filtering.

3- The visual recipients of motion can be modeld using oriented band-pass filters, such as Gabor filters. The analysis is done using a bank of filters as the following:

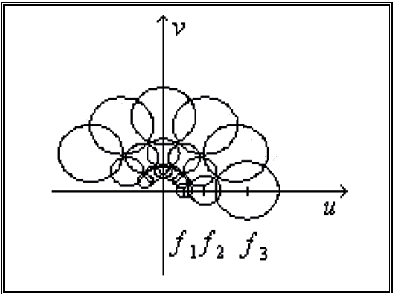


Fig.29 Gabor bank of filters

where f_1= *0,03125*, f_2= *0,0625* and f_3=*0,125*. All the images are analysed by each filter of the bank in order to perceive the real motion of the object and avoid the overture problem.

4- The resulting data is a vector field that shows the motion of each pixel, the module of those images is what is called dynamic saliency image. Finally, those images are temporary filtered using a 13 median filter.

After all this process the dynamic saliency image of the same frames as in Fig.25 is:

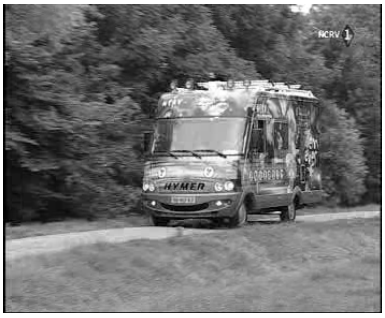
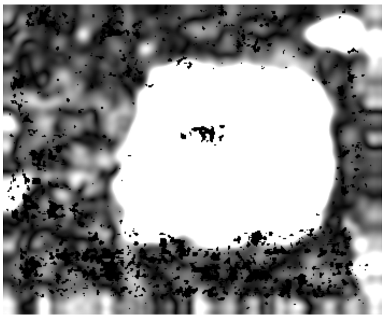
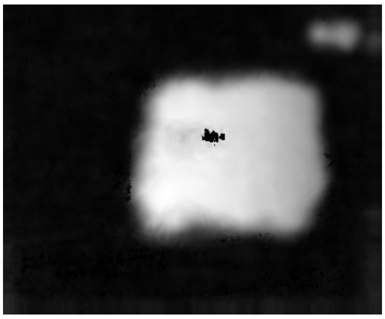
109	Saliency 109	Saliency 109 after temporal filtering
		

Fig.30 Original frame and its dynamic saliency images.

With this image it was also analysed the evolution of the centre of gravity, providing no satisfying results either.

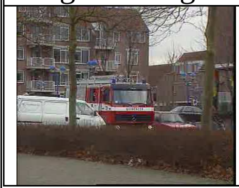
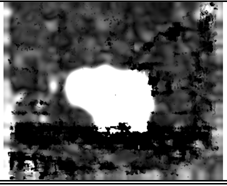
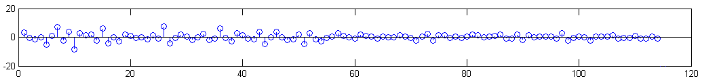
Original image	Saliency image	Evolution of centre or gravity
		

Fig.31 Horizontal evolution of centre of gravity in real sequence using saliency image

Analogously to the compensated images solution, the element in motion creates a different shape for each image, making the evolution analysis pointless.

Two patches of white pixels can be observed in the difference images shown in Fig.28 and in the saliency image in Fig.31:

- <u>Moving element(s)</u>: In this example the only element in movement is the van, and it is the element whose motion must be studied. In case there is more than one object in motion or the objects are deformable (like a person walking) some operations would be needed to obtain the motion of each of those elements.

- <u>Synthetically added objects</u>: Any object added a posteriori would fit in this category, like subtitles or the scoreboard of a football match. The channel logo can be seen in the top right corner of the example image in Fig.31. It is not a moving object, in fact it is not even in the scene, so it has to be identified and ignored as it can create "ghost activity" that is not real.

The barycentre is calculated using both elements, it is the barycentre of the whole image. The static element makes the centre of gravity to be always displaced to the non-moving element, making its evolution not to represent the real movement.

5.5 <u>K-Means</u>

In this work, K-Means is used to check the performance of the extracted features and the created histograms. K-Means is a popular clustering algorithm and one of the simplest *unsupervised learning techniques*. The procedure consists in classifying a given set of data into a certain amount of clusters fixed a priori (*k*). This method consists in:

- 1st: Place the k centroids of the clusters, different starting conditions can cause different results, in this case the centroids are the k first elements to be clustered.

- 2nd: Take each point belonging to a given data set and associate it to the nearest centroid. When all points are assigned an early grouping is done.

- 3rd: Using the previous results, centroids are recalculated trying to minimise an objective function, the most popular is a square-error function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \| x_i^{(j)} - c_j \|^2 \tag{14}$$

where $\|x_i^{(j)} - c_j\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the n data points from their respective cluster centres.

The second and third steps are performed repeatedly until the centroids are the same as the ones calculated the previous iteration. Here an example with 20 random points and 3 clusters:

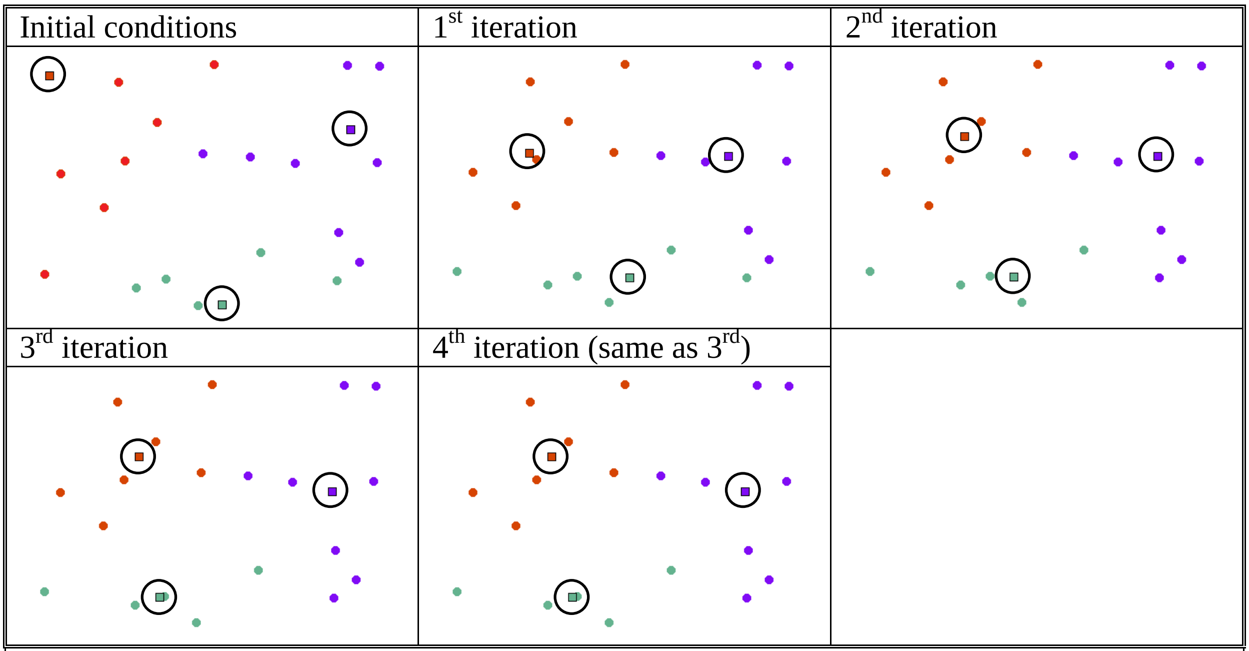


Fig.32 Evolution of k-means iterations

6. Development

6.1 Environment

Different tools were used for the work presented in this document. Fig.33 depicts the complete schema of the software used.

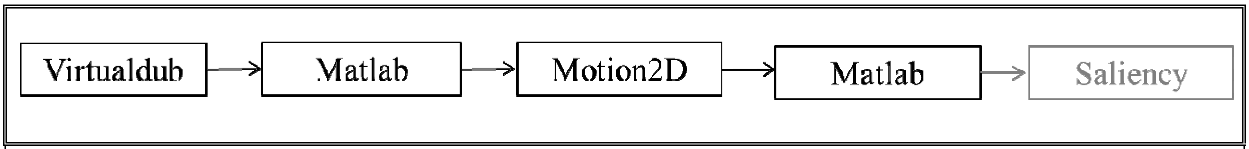


Fig.33 Block diagram of software tools

Virtualdub is used to extract images from an *mpeg* video, which are later used by Matlab. Motion2D is called by Matlab, it passes the images under analysis and receives the results. Finally, when the saliency of images was studied, it was necessary to use part of an existing program created by Sophie Marat[11].

VirtualDub

VirtualDub is a free software video capture/processing utility[12] that can extract some fragments of video and store them as short videos. It also has the option of exporting those fragments into a set. All images extracted are in *.bmp* format keeping original colour, and then MatLab makes a copy in greyscale and in *.png* format, and those are the images Motion2D works with.

[11] Complete Sophie Marat's Ph.D. [9] (in French)

[12] VirtualDub can be found in: http://www.virtualdub.org/

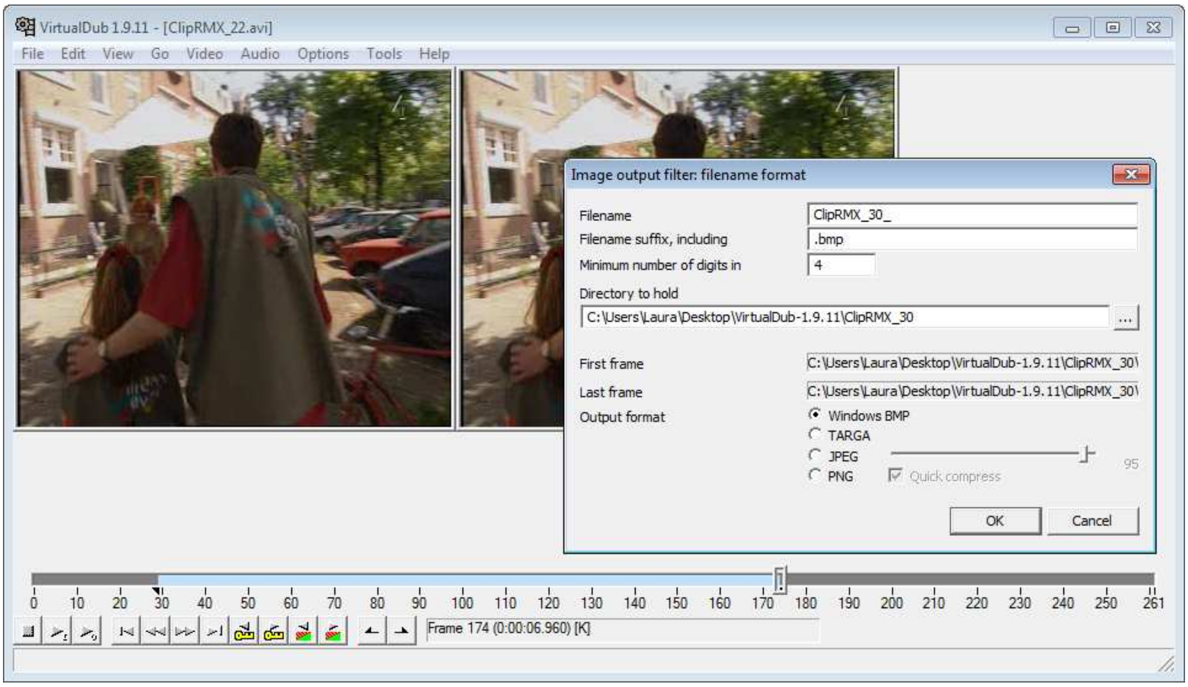


Image output filter: filename format

Filename: ClipRMX_30_
Filename suffix, including: .bmp
Minimum number of digits in: 4
Directory to hold: C:\Users\Laura\Desktop\VirtualDub-1.9.11\ClipRMX_30
First frame: C:\Users\Laura\Desktop\VirtualDub-1.9.11\ClipRMX_30\
Last frame: C:\Users\Laura\Desktop\VirtualDub-1.9.11\ClipRMX_30\
Output format: ● Windows BMP ○ TARGA ○ JPEG ○ PNG ☑ Quick compress 95

Frame 174 (0:00:06.960) [K]

Fig.34 Capture of VirtualDub when exporting as *.bmp* images

Motion2D

The software used to detect the camera motion is Motion2D, software created by the Vista team of Irisa and INRIA in Rennes that detects the dominant motion in a sequence. It can work with several types of images and also deal directly with MPEG-2 files. In this case the method chosen is working with simple images, extracted by *VirtualDub*.

This program also provides some other options: first frame, step, number of iterations,... All data resulting from the analysis are stored in a .txt file so they can be used later, or in other parts of the program. The most interesting part of that file is that it includes a key, so the user can always know which parameter is each number easily and it also says the type of estimation used, in this case “MDL_AFF_COMPLET”, which means complete affine model.

Table 8 contains an example of the file generated by Motion2D:

```
# Motion2D Copyright (c) 1995-2003 by INRIA
#
# This file contains the parameter values of the estimated 2D
# parametric motion model. A comment line starts with the #
# character. After the comments, the first line refers to the
# estimated model id. Next, each line refers to the motion
# model parameters estimated between two successive images.
#
# The data signification is given below.
#
# |------------------------------------------------------|
# | column | data signification for each estimation      |
# | number | between two successive images.              |
# |--------|---------------------------------------------|
# |   1    | number of the first image                   |
# |--------|---------------------------------------------|
# |   2    | motion model origin (row coordinate or yc)  |
# |   3    | motion model origin (column coordinate or xc)|
# |--------|---------------------------------------------|
# |   4    | motion model parameter (c1)                 |
# |   5    | motion model parameter (c2)                 |
# |--------|---------------------------------------------|
# |   6    | motion model parameter (a1)                 |
# |   7    | motion model parameter (a2)                 |
# |   8    | motion model parameter (a3)                 |
# |   9    | motion model parameter (a4)                 |
# |--------|---------------------------------------------|
# |  10    | motion model parameter (q1)                 |
# |  11    | motion model parameter (q2)                 |
# |  12    | motion model parameter (q3)                 |
# |  13    | motion model parameter (q4)                 |
# |  14    | motion model parameter (q5)                 |
# |  15    | motion model parameter (q6)                 |
# |--------|---------------------------------------------|
# |  16    | illumination variation parameter            |
# |--------|---------------------------------------------|
# |  17    | support size (only if computed, by default) |
# |--------|---------------------------------------------|
#
MDL_AFF_COMPLET
0 233.500000 350.000000 0.293268 -0.000819 -0.000614 0.001586 0.000036
-0.000203 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.047634
0.894448
1 233.500000 350.000000 0.259153 -0.022124 -0.000531 0.001412 0.000082
-0.000290 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.043557
0.894203
2 233.500000 350.000000 0.263665 -0.007318 -0.000514 0.001447 0.000044
-0.000228 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.043814
0.894221
3 233.500000 350.000000 0.270709 0.005278 -0.000501 0.001494 0.000017
-0.000159 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.043285
0.894307
```

Centre point of the image

Parameters not calculated in the model used

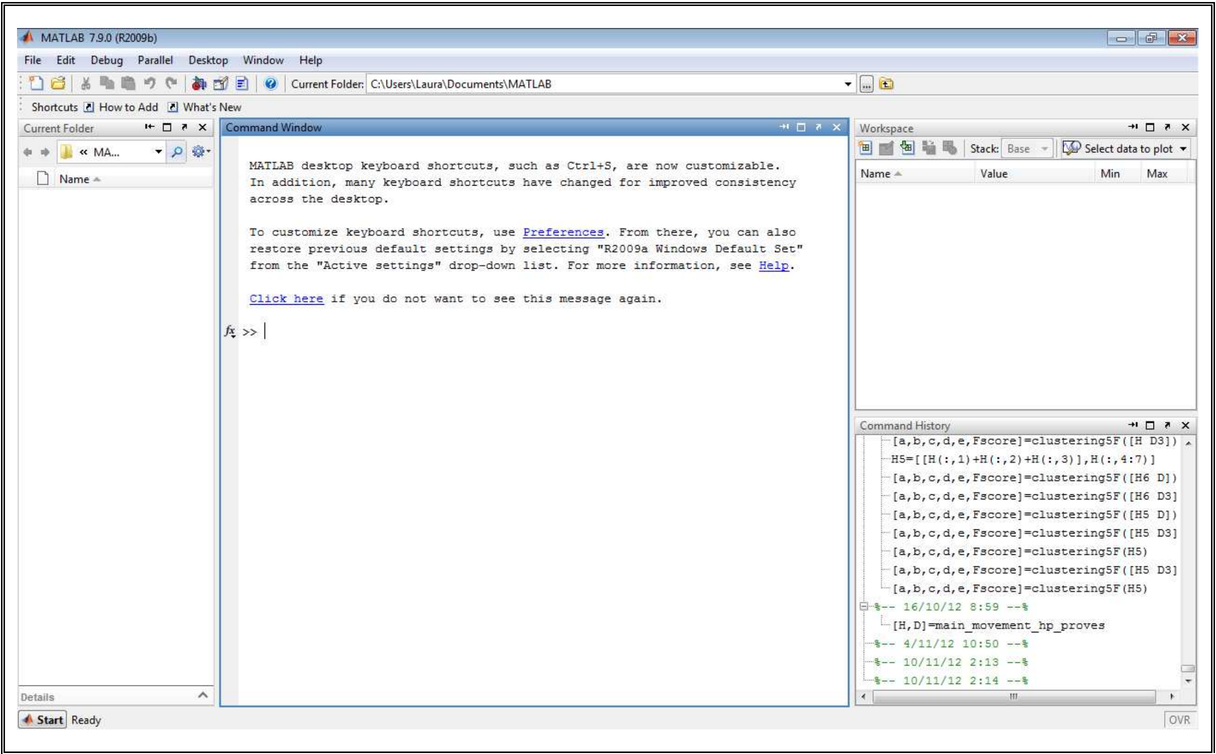
(…)

Table 8 Fragment of the resulting file of Motion2D

Matlab

Matlab (Matrix Laboratory) is a popular numerical computing environment with its own language (language M). Created by the company Mathworks, it allows matrix manipulation, function and data plotting, interfacing with programs in other languages such as C or JAVA, and many other features that were not needed for the work presented in this document.

The first application of Matlab tools is the conversion of the colour *.bmp* images into black and white *.png*. Virtualdub does not have the option of greyscale images in 2D matrices, so it is necessary for Matlab to convert the 3D colour matrices into greyscale images suitable for the following steps. After the images are converted and stored for future experiments with the same dataset, MatLab calls Motion2D to analyse the camera motion. Results provided by Motion2D are used to create the histograms of motion and the compensated images. Finally, MatLab calculates the distances between shots, performs the clustering and evaluates the results, as presented in next Chapter 7.



```
MATLAB desktop keyboard shortcuts, such as Ctrl+S, are now customizable.
In addition, many keyboard shortcuts have changed for improved consistency
across the desktop.

To customize keyboard shortcuts, use Preferences. From there, you can also
restore previous default settings by selecting "R2009a Windows Default Set"
from the "Active settings" drop-down list. For more information, see Help.

Click here if you do not want to see this message again.

fx >> 
```

```
[a,b,c,d,e,Fscore]=clustering5F([H D3])
H5=[[H(:,1)+H(:,2)+H(:,3)],H(:,4:7)]
[a,b,c,d,e,Fscore]=clustering5F([H6 D])
[a,b,c,d,e,Fscore]=clustering5F([H6 D3]
[a,b,c,d,e,Fscore]=clustering5F([H5 D])
[a,b,c,d,e,Fscore]=clustering5F([H5 D3]
[a,b,c,d,e,Fscore]=clustering5F(H5)
[a,b,c,d,e,Fscore]=clustering5F([H5 D3]
[a,b,c,d,e,Fscore]=clustering5F(H5)
%-- 16/10/12 8:59 --%
[H,D]=main_movement_hp_proves
%-- 4/11/12 10:50 --%
%-- 10/11/12 2:13 --%
%-- 10/11/12 2:14 --%
```

Fig.35 Capture of MatLab default desktop

Saliance analysis

Operations performed for the saliency analysis were implemented in Sophie Marat’s PhD studies static saliency, dynamic saliency and includes a face detector. The three tools are combined to detect the places where the human eye centres the attention. For the work presented here, the only part used was the dynamic saliency.

6.2 Databases used

During this project different types of videos, were used which can all be separated in three categories: synthetic, films and documentary. Synthetic sequences were used for the firsts tests to characterise and evaluate the performance of Motion2D, using sequences which camera motion was already knew was necessary to find the problematic points where Motion2D did not have good results. The sequences extracted from films where used to find the different bins sizes, as in films scenes are prepared; the camera motion is clean and regular.

TrecVid images, the documentary, were used not only to be able to compare results with other laboratories, they also offer the opportunity to start thinking of taking part of it the following years. Furthermore, documentaries are not planned and the motion is much more difficult to be compared, as it is more irregular.

Synthetic sequences

These images were the ones to start, instead of having sequences from the real life, they were images displaced manually to make an effect of travelling, amplified to create the effect of zoom in or reduced for zoom out. There were some clear images, blurry images and also some sequences with two elements: a blurry background and an object in foreground both moving separately. All those images were created specifically for this project, and they worked out to be very useful to evaluate the performance of the system, as the results were already known.

Fig.36 Some examples of synthetic images used: two clear images, one blurry image, and two with moving elements

Hollywood

In [8] the database used is called *Hollywood Human Actions dataset*, and it contains sequences of 32 movies where some common actions are performed (get out of a car, stand up, shake hands …). This database is only 2,4 GB, but there is an extension (*Hollywood-2 Human Actions and Scenes dataset*) with 40 GB of films.

In this work the database used has been *Hollywood*, as it is a small database and it has been enough to analyse the performance of Motion2D. It is very complete, as some shots are extracted from old films in black and white; there are some scenes with handheld camera, fast action scenes… all kind of situations to be analysed in terms of camera motion.

The clips used are fragments belonging to colour shots. They were cut even more as some shots were too long to work with them, or there was a change of camera and that was a problem for Motion2D, as it is not prepared to detect that. Below some frames of the used videos.

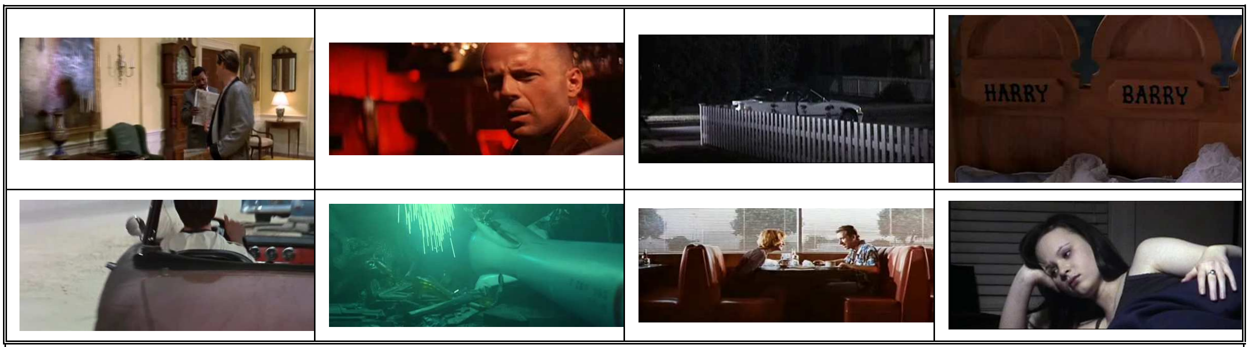


Fig.37 Some examples of *Hollywood* database videos used.

Documentary

In 2008 the Netherlands Institute for Sound and Vision prepared a dataset of videos containing news magazine, science news, news reports, documentaries, educational programming and archival in MPEG-1 for use within *TrecVid*[13]. The videos are between 15 and 30 minutes long and they all present the same definition: 352x288 pixels, unlike the two other previous sets of images, where each video has different sizes and aspect ratio.

Only some fragments of 30 seconds have been extracted, as longer videos were very slow to process with Matlab, so the resulting clips are 30 short videos that could be classified into three categories.

6.3 Implementation

The algorithms used in this project can be classified in three groups:

- Previously existing source code: All the part referring to the saliency detector was written by Sophie Marat. This previous work was exploited by calling her functions with the suitable parameters.

  The clustering algorithm was also created by another person[14], and it was just readjusted to fit the algorithm's requirements.

- Functional code: Part of the developed code aimed at connecting the existing pieces to evaluate the video clustering strategies. This wrapper was the responsible of loading images, call the functions, make the histograms...

- Evaluation functions[15]: Implementation of the experiments necessary to evaluate the proposed design.

[13] *TrecVid*: Evaluation campaign whose goal is encouraging the research in information retrieval by providing a large test collection, uniform scoring procedures and a forum for organisations interested in comparing results.

[14] Kandi Teknomo's website: http://people.revoledu.com/kardi/tutorial/kMean/matlab_kMeans.htm

[15] The commented code of those functions can be found in Annex [C].

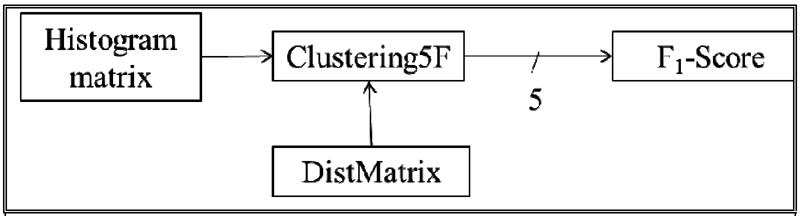


Fig.38 Block diagram of evaluation steps

- Clustering5F: This function gets the matrix with the values of all histograms being a clip in each row. The rows are randomised for cross-validation and then the K-Means algorithm makes the clusters and calculates the F_1-Score. There are five iterations each time the function is called; it returns the value of each iteration and finally the mean value of them.

- Fscore: Function to calculate the F_1-Score automatically, as in clustering each pair has to be evaluated, it is essential to have a short program to calculate it. The program receives two vectors, one with the clustering results and the other with the Ground-Truth. The function calculates pairs of the True Positives, False Negatives, False Positives and True Negatives. The Precision, Recall and F_1-Score are computed based on these figures.

- DistMatrix: Originally created by the same author as the K-Means function, it is used to calculate the distances between the elements of the cluster and the centroids. It receives two matrices and returns another matrix with the distances between all the elements and the centroids. This function originally calculated other distances, so following the original function Euclidean, Manhattan and Bhattacharyya were added.

7. Evaluation and Results

The performance of the different configurations of the system was evaluated using the set of 30 fragments of video from TrecVid explained in the previous chapter. The dataset was prepared to have four differentiated groups shown in Fig.39.

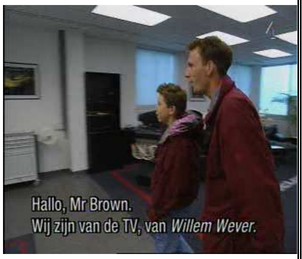
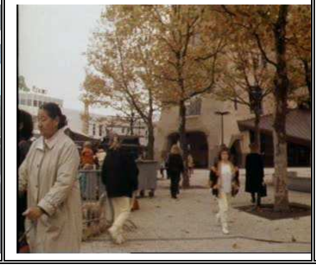
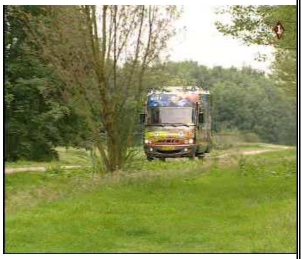
Human activity		Unanimated	
Static	Moving	Static	Moving
Bust of a person talking, different races and situations.	Two or three people walking followed by the camera.	Travelling in a static scenario, or with a little activity.	Camera following a plane or a van.
			

Fig.39 Some examples of TrecVid database videos used.

Preliminary experiments showed that categories where moving elements are followed by the camera are too similar to be in two different groups, so the final experiment was done with only three categories:

- Person talking: Most of clips are the bust of a person, but there are some that give a wider shot and some others that have subtitles. In all cases the camera remains still or almost, as some scenes are recorded using handheld camera.

- Tracking: A camera follows a person or more, or a vehicle. The camera moves slowly following the elements of interest.

- Travelling: The camera sweeps a scene to show it completely. The camera moves faster than in Tracking as there are no elements moving. This kind of shots is usually used to give a general view of the situation.

Preliminary results

The first evaluation of the system was a *naked eye* analysis of the distance matrices. Histograms of all clips were compared using Euclidean (L2) and Bhattacharyya distance, and results were organized in a 30x30 matrix where the *travelling* shots are shown in yellow, the *person talking* shots in green and the *tracking* ones in blue. The histograms comparison shown in Table 9 and Table 10 correspond to the 7 horizontal bins and 5 bins divergence[16].

L2 Distance Horizontal Translation									
	1	3	7	12	18	6	15	26	30
1		0,29	0,33	1,3	1,3	1,21	0,82	0,67	0,88
3			0,31	1,17	0,17	1,07	0,62	0,44	0,69
7				1,01	1,01	0,91	0,56	0,44	0,57
12					0	0,18	1,04	0,97	0,8
18						0,18	1,04	0,97	0,8
6							0,9	0,85	0,67
15								0,24	0,3
26									0,34
30									

Table 9 L2 distance of horizontal travelling histograms (fragment)

The black cells show the confusing results while the cells in colour highlight the distances between elements of the same category. As distances are commutative, the table does not include redundant information to keep it as simple as possible.

[16] The histograms of the shots whose comparison is shown in Table 9, Table 10, Table 11 and Table 12 can be found in Annex [D]

Bhattacharyya Distance Horizontal Translation

	1	3	7	12	18	6	15	26	30
1		0,02	0,04	0,77	0,77	0,72	0,16	0,1	0,25
3			0,08	0,6	0,6	0,55	0,11	0,05	0,19
7				0,36	0,36	0,29	0,11	0,08	0,14
12					0	0,03	0,55	0,42	0,28
18						0,03	0,55	0,42	0,28
6							0,37	0,33	0,19
15								0,01	0,05
26									0,06
30									

Table 10 Bhattacharyya distance of horizontal travelling histograms (fragment)

Although the matrices show only a portion of what is calculated, it can be seen that L2 provides more differentiated results between what is similar and what is not similar, while Bhattacharyya's results are more continuous, there is not a such a clear frontier.

If comparing divergence matrices, results point that divergence cannot be used to make the clustering all by itself, but it could be useful to refine some elements that are difficult to cluster.

L2 Distance Divergence

	1	3	7	12	18	6	15	26	30
1		0,43	0,19	0,02	0,01	0,02	1,1	0,92	1,02
3			0,38	0,44	0,42	0,44	0,71	0,52	0,59
7				0,21	0,2	0,21	0,98	0,82	0,94
12					0,01	0	1,12	0,94	1,03
18						0,01	1,1	0,93	1,02
6							1,12	0,94	1,03
15								0,2	0,28
26									0,17
30									

Table 11 L2 distance of divergence histograms (fragment)

Bhattacharyya Distance Divergence									
	1	3	7	12	18	6	15	26	30
1		0,07	0,02	0,003	0,001	0,003	0,36	0,22	0,25
3			0,12	0,08	0,06	0,08	0,14	0,06	0,05
7				0,03	0,03	0,03	0,3	0,2	0,28
12					0,002	0	0,47	0,28	0,29
18						0,002	0,36	0,21	0,23
6							0,47	0,28	0,29
15								0,01	0,03
26									0,01
30									

Table 12 Bhattacharyya distance of divergence histograms (fragment)

Using TBM this refining process could be done, unfortunately it was impossible to try it within the internship, so a simpler algorithm was used to evaluate the performance of the technique under study.

F_1-Score

F_β-Score uses the Precision (P) and Recall (R)[17] to measure the accuracy of the system. β can be adjusted to give more weight to one or another and then penalise false positives (β>1) or true negatives (β<1). In the F_1-Score case β=1, so they have the same importance.

- Precision: Fraction of relevant documents retrieved over the number of documents retrieved. In clustering it is the fraction of correct pairs over the total number of pairs made.

$$P=\frac{TP}{TP+FP} \tag{15}$$

- Recall: Fraction of retrieved documents over the number of relevant documents existing. In clustering it is the fraction of correct pairs over the correct pairs made and the elements clustering in different groups that should have been in the same one.

[17] An article about results evaluation was published in the bitseach blog. It can be found in Annex [A], Table

$$R=\frac{TP}{TP+FN} \qquad (17)$$

Precision and Recall are often presented combined in a Precision-Recall graph. F_1-Score combines those two measures in a new one, so instead of a graph a numeric table can be created.

$$F_\beta=\frac{(\beta^2+1)PR}{\beta^2P+R}=\{\beta=1\}=\frac{2\mathrm{PR}}{P+R} \qquad (15)$$

In all three equations:

- TP: True Positive, two similar elements assigned to the same cluster
- FP: False Positive, two dissimilar elements in the same cluster
- TN: True Negative, two different elements assigned to different clusters
- FN: False Negative, two similar elements in different clusters.

The clustering was done using 5, 6 and 7 horizontal bins and 3 o 5 for divergence. There was also an analysis without using divergence histogram, so it were possible to check if it really improved results or not.

Results in Table 13 were obtained with K-Means making the clustering of the set of 30 clips. Using different configurations of histograms and different distances 5 iterations were performed for each configuration. The sorting of the input elements for the K-Means algorithm affects the results, so the 5 iterations received the elements in random order and F_1-score calculated for each experiment. Table 13 contains the averaged F_1-scores[18].

[18] The results of each iteration can be found in Annex [E].

	L1	L2	Bhattacharyya
5H	0,58	0,58	0,53
5H 3D	0,55	0,59	0,59
5H 5D	0,57	0,6	0,56
6H	0,64	0,65	0,56
6H 3D	0,59	0,64	0,61
6H 5D	0,58	0,61	0,57
7H	0,55	0,59	0,57
7H 3D	0,55	0,59	0,56
7H 5D	0,56	0,64	0,57

Table 13 F_1-score for different configurations and distances

Observing the results, it is clear that L2 is the distance that obtains better results, although L1 also interesting. Bhattacharyya distance presents the worst results. As L1 and L2 are calculated using a very similar method, it is not strange that they provide similar results as well. L1 and L2 calculate the difference between the components of the vectors, while Bhattacharyya multiplies the components, making small coincidences more important than in the other cases.

In the work done by H. Göeau [2] colour and orientation had 64 and 32 bins respectively, in those cases Bhattacharyya was the best option, but in this case histograms have, at maximum, 12 bins. The fact of having much less bins, makes small coincidences useless or even confusing, so simpler distances work much better in this case.

8. Conclusions

In this project, it has been shown how it is possible to discriminate videos depending on their semantic content only using the motion of the camera. This is possible because the camera motion techniques may be related to the type of scene that is being filmed.

Different histogram configurations have been tested, as well as different distances in order to find the combination that obtains the best results. Experiments have shown that, for histograms that have few bins, it is essential to detect big coincidences. Bhattacharyya distance offered the worst results because it multiplies the values of the bins to find the distance between them, making small coincidences important. Between L1 and L2, there is not much difference, but the common L2 distance generated better results with the working dataset.

The number of bins has also provided different results, although the most important part of bins definition is their sizes, being non-uniform and relative to the frame dimensions, makes them more adaptive to different types of clips. Having small bins for slow motion and bigger ones when the motion is faster, makes it possible to study slow motion and “static motion” much deeply. Having too much bins can be counterproductive, as it can create differences between clips that would be perceived similar. For example, the configuration of 7+5 bins uses too many bins to represent slow motion and zoom.

In the presented experiment clips were chosen to have different types of motion, but two categories were merged in a single *tracking* class, as the camera motion technique is the same whether it is following people or vehicles. Additional features, such as face or object detectors could have made the discrimination of the two merged categories possible.

Analysing the motion of the elements in the scene at a local scale could also give a lot of information and be very helpful when classifying videos, but it is not an easy task. In this project there was an attempt to find an easy way of doing it, but modelising elements in the scene usually requires segmentation and tracking techniques that fall out of the scope of this work. The most important problem was given by the constant change of shape of the object in the scene, as well as the variation in appearance generated by light changes. If the element to be studied does not keep its visual features it is difficult to track it.

In GIPSA-lab the objective is create a multi-feature analysis to combine the static features, camera motion, elements motion and sound. All information brought by the different analysis will be fused using the Transferable Belief Model.

Future tasks related to the work presented in this project are:

- Vertical motion. The vertical motion is not usual, but a clear motion upwards or downwards would be very characteristic for a clip. Furthermore, the detection of vertical shaking (explained in Fig. 7) could make it possible to discriminate between handheld and “tripod” clips, being the former more common in action scenes or homemade films and the later ones more common in static scenes.

- Element’s motion: The rejected ideas to model the motion of the elements that appear in the scene did not provide good results, but the fact of being able to compensate the camera motion is the first step to model the activity in the scene.

- Transferable Belief Model: The aim of this technique is combining different types of information coming from different sources and try to make the most of it. Information coming from each source can be treated differently, so divergence could be used for refining doubtful results, while the clear ones would not need any more processing. The semantic interpretation could be improved by fusing the static image features (colour and texture), with the camera motion. For example, clips with a vehicle could be discriminated from the clips with people walking, as cars and vans have a much more rectangular shape.

Finally, just point out that concepts related to artificial intelligence and machine learning, are much more complicated than what has always been presented in TV or cinema. When someone really must go deeply in these topics for the first time, it is always disappointing, as it never reaches the expectations that science-fiction films have created. Nevertheless, this field of study is improving and growing every day, so maybe in a few years it will be possible that a computer classifies multimedia data as any person would do.

9. References

1. D.G.Lowe, *Object Recognition from Local Scale-Invariant Features*, International Conference on Computer Vision, Corfu, Greece 1999

2. Göeau,H. *Structuration de collections d'images par apprentissage actif crédibiliste,* (2009)

3. Göeau,H.; Rombaut,M.; Pellerign,D.; Buisson,O. *Multi-labeled image classification by TBM active learning*, Workshop on Theory of Belief Functions, Brest : France (2010)

4. Gorelick,L.; Blank,M.; Shechtman,E.; Irani,M.; Basri, R. *Actions as space-time shapes*, Tenth IEEE International Conference on Computer Vision (ICCV), 2005

5. Guironnet,M.; Pellerin, D.; Rombaut, M. *A fusion architecture based on TBM for camera motion classification*, Image and Vision Computing 25, 11 (2007)

6. Junejo, I.N.; Dexter,E.; Laptev, I.; Pérez, P. *View-Independent Action Recognition from Temporal Self-Similarities,* Pattern Analysis and Machine Intelligence, March 2010

7. Laptev, I.; Lindeberg, T.; *Space-time interest points*, ICCV'03

8. Laptev, I.; Marszalek, M.; Schmid, C.; *Rozenfeld, B. Learning realistic human actions from movies*. Computer Vision and Pattern Recognition, 2008 (CVPR 2008)

9. Marat,S. *Modèles de Saillance visuelle par fusión d'informations sur la luminance, le mouvement et les visages pour la prédiction de mouvements oculaires lors de l'exploration de vidéos,* (2010)

10. Niebles, J.C.; Chen, C-W; Fei-Fei, L. *Modeling Temporal Structure of Decomposable Motion Segments for Activity Classification*. Proceedings of the 12th European Conference of Computer Vision (ECCV). 2010

11. P.Smets. *Belief Functions and the Transferable Belief Model*, Documentation Section on the website of the Society for Imprecise Probability Theory and Applications (SIPTA)

12. Robertson, S.; *Understanding Inverse Document Frequency: On theoretical arguments for IDF*, Journal of Documentation, 60(5) (2004)

13. Rubner,Y.; Tomasi,C.; Guibas,L.J. *The Earths Mover's Distance as a Metic for Image Retrieval*, International Journal of Computer Vision 40(2), 99-121, 2000

14. Simac-Lejeune, A.; Marat, S.; Pellerin, D.; Lambert, P.; Rombaut, M.; Guyader, N. *Relevance of interest points for eye position prediction on videos*. Computer Vision System. 7th International Conference on Computer Vision Systems, ICVS 2009

15. Sivic, J, Zisserman, A.; *Video Google: A Text Retrieval Approach to Object Matching in Videos,* ICCV'03

10. Annexes

A. Short articles

In this section there can be found some articles written before the beginning of the project that are related to the work presented in this document. The articles are published in Bitsearch[19], a blog with the work of Telecommunication Engineering students that are related to the image group in UPC Barcelona.

Retrieval Systems Evaluation

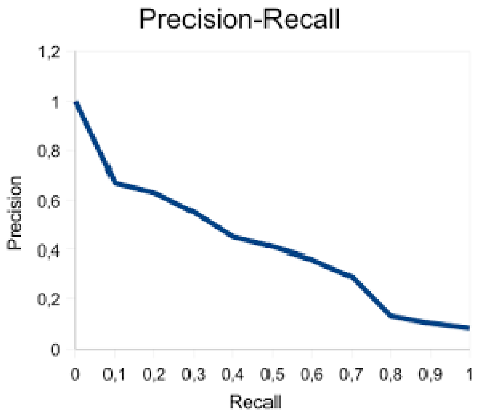
In retrieval systems it is very important to use some measures to evaluate how good is the system that is being used, so it can be compared with others. What is usually evaluated is the performance of the system when the user asks a query and the system retrieves a group of images from the database. The most popular systems are precision, recall and F-score; but the last one is just a mix of the previous both.

precision=#relevant retrieved docs/#retrieved docs
recall=#relevant retrieved docs/#relevant docs

These two measurements are often represented in a single diagram called precision-recall curve.

F-score or F-measure allows the user to give weights to precision and recall using this expression:

Fβ = (1+β^2)·precision·recall/(β^2·precision+recall)

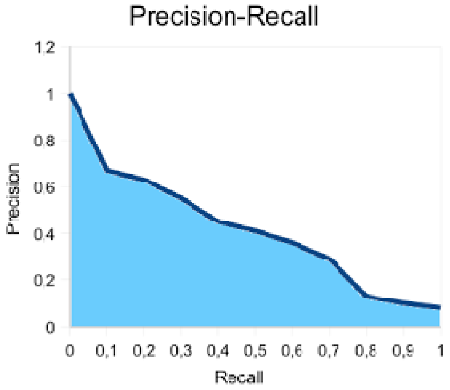


All this measures do not care about the order the results are given, so if it is wanted to evaluate the system from that perspective other methods must be used.

[19] http://bitsearch.blogspot.com.es

Mean Average Precision (MAP)

This is the evaluation method used in TRECVID; furthermore MAP also approximates the average area under the curve precision-recall for a set of queries. It calculates the average precision of every single query and then the mean value of all queries.



Considering:

Q: #queries

mj: #relevant documents in the database for query j = #max relevant results for query j

Rjk= #relevant results of query j until k position

The expression to calculate it is:

$$\mathrm{MAP}(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\mathrm{Precision}(R_{jk})$$

Averaged Normalized Modified Retrieval Rate (ANMRR)

The ANMRR is used by the MPEG group to evaluate the performance of a system. Its interesting point is the penalisation for missing relevant results as it follows:

$$\mathbf{Rank}(k) = \begin{cases} \mathbf{Rank}(k) & \text{if} \quad \mathbf{Rank}(k) \leq K(q) \\ 1.25 \cdot K(q) & \text{if} \quad \mathbf{Rank}(k) > K(q) \end{cases}$$

Rank(k) = position in which relevant image k is retrieved

K(q) = #images retrieved for query q. It is defined using this equation where NG(q) is the number of relevant images in the database.

$$K(q) = \min\{4 \cdot NG(q), 2 \cdot \max[NG(q), \forall q]\}$$

After redefining the Rank, the Average Rank is calculated:

$$\mathbf{AVR}(q) = \frac{1}{NG(q)}\sum_{k=1}^{NG(q)}\mathbf{Rank}(k)$$

In order to eliminate the influence of different NG, the MRR (Modified Retrieval Rank) is defined:

$$\mathbf{MRR}(q) = \mathbf{AVR}(q) - 0.5 \cdot [1 + NG(q)]$$

To get a value between 0 and 1 it is necessary the NRRM (Normalised Modified Retrieval Rank):

$$\mathbf{NMRR}(q) = \frac{\mathbf{MRR}(q)}{1.25 \cdot K(q) - 0.5 \cdot \lceil 1 + NG(q) \rceil}$$

Finally, to obtain the average of several queries the ANMRR (Average Normalised Modified Retrieval Rank) uses this expression where NQ is the number of queries done.

$$\mathbf{ANMRR}(q) = \frac{1}{NQ} \sum_{q=1}^{NQ} \mathbf{NMRR}(q)$$

Rank

This measure system is the one used by the authors of “Video Google”. It is calculated using the expression below where:

$$\widetilde{Rank} = \frac{1}{NN_{rel}} \left(\sum_{i=1}^{N_{rel}} R_i - \frac{N_{rel}(N_{rel}+1)}{2} \right)$$

N: #images in the database
Nrel: relevant images
Ri: rank of the ith relevant image.

This equation will return 1 if all relevant images are returned first and 0,5 in a random retrieval, 0 would mean that all relevant results have been retrieved in last positions.

Table 14 Article written in bitsearch blog, published in March 2010

A summary of the Video Google paper by Sivic and Zissermann

Video Google: A Text Retrieval Approach to Object Matching in Videos, this is the title of a popular paper by Josef Sivic and Andrew Zisserman where they explain how methods used by Google for text retrieval can be applied to images. As it is a dense document, we thought that it could be very useful to have a brief summary of the article in our blog so everyone interested will not need to read the whole article.

The aim of the work was to retrieve keyframes with certain objects or locations with the speed and accuracy provided by Google when retrieving text documents. To do that it is necessary to define some regions in the image and create a visual vocabulary or codebook with them so visual words can be treated like text words are treated by Google.

Defining regions

Two methods are used to detect two types of regions in a frame:

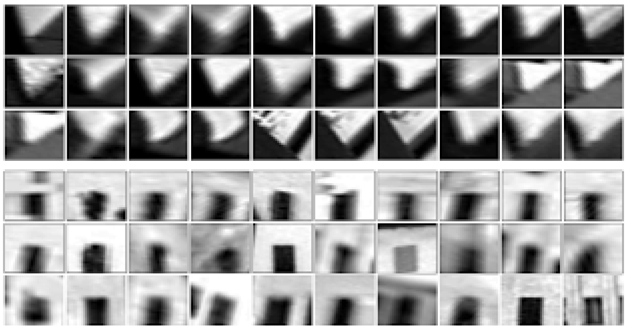
- Shape Adapted (SA): This method detects corner-like features by iteratively determining the centre, scale and shape of the ellipse that surrounds a point of interest.

- Maximally Stable (MS): To find regions with a high contrast with their surroundings the procedure is an intensity watershed image segmentation, and the regions that keep their area after an intensity threshold variation are the ones to work with.

Both algorithms define some elliptical regions on every keyframe, and each region is represented by a 128-dimensional vector using the SIFT descriptor.

There is tracking over a sequence of frames to avoid having unstable regions. If a region doesn't last for three frames at least, it is rejected.

Visual vocabulary



To build our "vocabulary" we need to quantize the descriptors into clusters, and each cluster will represent a "visual word". In this case the clusters are generated using the K-means algorithm, but other methods could have been used. The K-means algorithm is run several times setting the centre of the clusters randomly and then the best result is used obtaining about 6000 clusters are used for SA regions and 10000 for MS. The number of clusters is chosen empirically to maximize the retrieval results. The picture in the left shows two clusters from SA regions.

While a text document can be characterised with a vector of word frequencies, the same happens on images with the obtained visual words. In text retrieval it is a very common practice to assign weights to the words, so the analogue procedure is done with the visual words using the tf-idf algorithm, applicable for both text and image.

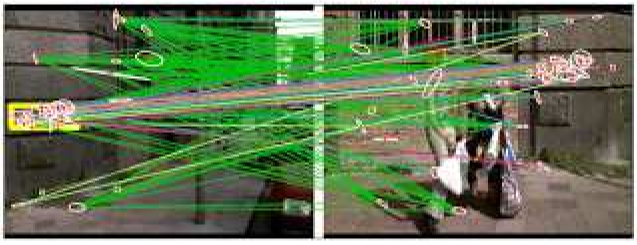
Experimental evaluation

Two kinds of experiments were run using the described methodology: scene matching and object retrieval.

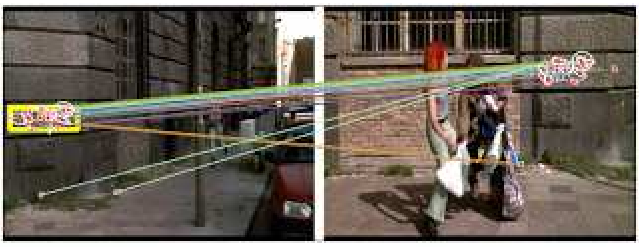
In scene matching the experiment consisted of matching a scene location from a group of frames. He have an example in the picture on the right, MS regions are in yellow and SA in blue. The corresponding regions are enough to determine that in both frames we have the same location.

The retrieval performance was measured using Rank, explained more detailed in the previous blog post on Retrieval Systems Evaluation.

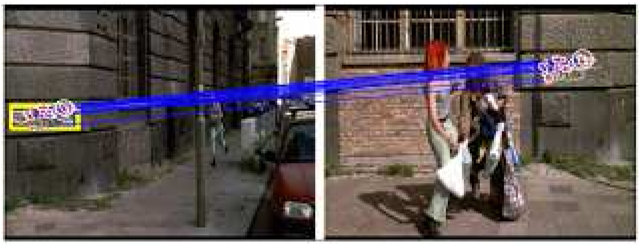
In object retrieval experiment the object of interest is selected by the user as a part of the frame. To improve the performance in object retrieval a few improvements were applied. Here we can see the original results.



- Stop List: To avoid mismatches and reduce the number of words in the codebook, the top 5% and the bottom 10% are deleted, so most common words and the very rare are rejected. Here we can see the results after applying the stop list.



- Spatial consistency: In a first stage results are obtained only using the weighted vector. After that results are re-ranked following the criteria of having the neighbouring matches into a surrounding area. After applying the last improvement there is only one result left, the correct match.



- Inverted files: The inverted file is a very useful tool for fast document retrieval. It is a list with all the words that appear in all the documents stored and then all the occurrences of the word in all the documents. In the experiment, the traditional text words were replaced by the visual ones.

In object retrieval the visual vocabulary created from one movie was used to find some objects in another film and the results were quite good, even for the second film. There were no false negatives and the false positives were in the bottom part of the ranked list, so precision was good. Thanks to the inverted file, results were fast, too.

Summary and conclusions

The analogy with text retrieval and image retrieval has been proved to be useful, but some improvements could be done. According to the authors low rankings are due to a lack of visual descriptors for some scene types, and in object retrieval it would be interesting to define the object from more than one frame to get more correct results. They also point out that using latent semantic indexing, like pLSA and automatic clustering could find the principal objects that occur in a movie.

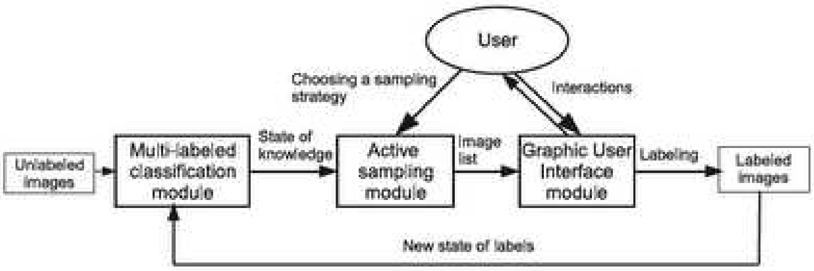
Table 15 Article written in Bitsearch blog, published in August 2010

First contact with Active learning with the Transferable Belief Model for Image Annotation

This February I will go on Erasmus to Grenoble, to the "Grenoble INP", in the Gipsa Lab (Grenoble Images Parole Signal Automatique) in the Image et Signal department. I have been proposed to work in a project of active learning with M. Rombaut and D. Pellerin. My first task has been reading the paper Multi-labeled image classification by TBM active learning, presented in the Workshop on Theory of Belief Functions (Belief 2010), Brest : France (2010).

When I was collaborating with the TSC department I became familiarized with supervised learning, as it was the system used in experiments such as Formula 1 publicity. This article is not about supervised learning, it is about active learning, a form of supervised learning that requires the interaction of the user. There are situations when we have some many images to annotate that any help can be useful. For example, in the F1 experiment we annotated the images randomly, but a little help to know which ones were the most useful could have provided better results with less annotated images.

This system is a tool to make user's work easier. Already labeled images in the database are used to structure the unlabeled ones and propose a suitable class for them. This assistance classification system selects images for the user which are interesting to classify according to a specific strategy and propose a label.



The framework is divided in two parts: an automatic part to "model the knowledge" and another part that requires the interaction of the user. Here we have a representation of the system, which is presented as three modules. As we can see, the first two modules are the ones where no interaction of the user is required, and the ones I will explain.

The first step consists in **modelling the knowledge** of the labeled images to predict the relevant label of the current unlabeled image. The knowledge is modelled using the following techinques.

- <u>Neighbour images</u> (a single one or K nearest neighbours): If image u is close to a classified image (or a set) there is a high believe that it will be labeled the same.

- Knowledge from all classes: Depending on the semantic interpretation of the images one image can be associated to several non-exclusive classes.

- Knowledge from all characteristics: This method allows to detect new classes or a new modality of a known class.

The second step is the **active sampling** module, where different criteria are presented to select the images that will be labeled by the user:

- Most positive unlabeled images: Sometimes names "most relevant". Here the system chooses the images that are easy to classify because the visual content is very similar to already labeled images.

- Most ambiguous unlabeled images: This strategy consists in choosing the unlabeled image which is on the limits of all the known classes. It can also be interesting to select images that are locally most ambiguous, the ones that are on the borders of a certain class.

- Most rejected unlabeled image: The selected images are the unlabeled ones that do not correspond to any class. They have been classified as not belonging to any of the already existing classes, so this can be useful to create a new class or correct an already existing classes.

- Most conflicted unlabeled image: The information fusion with all characteristics can lead to a conflict about the inclusion in one or more classes, so they might not correspond to current known classes.

- Most uncertain unlabeled image: The two hypothesis (belonging or not belonging to the class) have similar probabilities, so it is impossible to distinguish one hypothesis from the others.

The third step is the final classification of the selected images.

Table 16 Article written in Bitsearch blog, published in November 2011

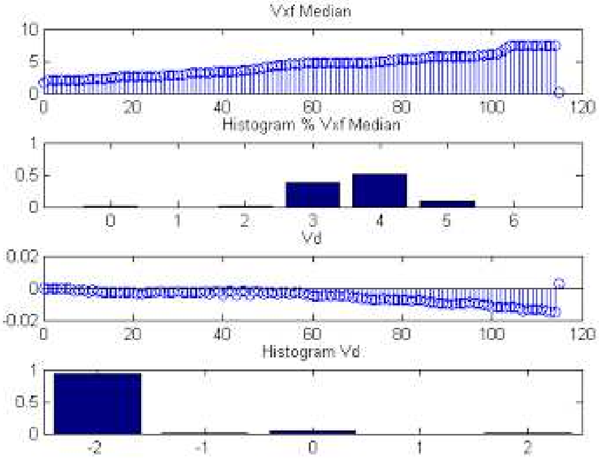
Camera Motion for Video Clustering

After 5 Months in Grenoble I am writing the report to be presented in Barcelona about my work there. Before leaving I wrote a short article about the system they already had to work with images, so now I will explain the work that I have done there.

The main idea of the team was to extrapolate a system originally meant to classify and annotate static images to be suitable for videos too. In my case I studied the camera motion as a way to characterise the videos, but there was also another student working with the sound and I am sure that other students will be working in that project extracting different features.

What I have been doing is characterise the videos by its dominant motion, that most of times it the motion of the camera. According to the type of action that happens in a scene the use of the camera can be very different. A simple example: if there is an element to be followed the camera will move according to that element, but if there is only a landscape to show and there is no element to follow, the camera would usually move faster, as there is nothing the observer has to pay special attention.

To extract the dominant motion of the scene the university provided me with Motion2D, a program created by Irisa that could detect the motion of the whole scene and then calculate the values of that motion according to different models. After a few tries we decided to work only using 2 values: horizontal motion and divergence (what we identify as zoom), as those two are the most usual movements to be performed by a camera.



After the motion is detected by Motion2D this information is represented in two histograms, one for horizontal motion and the other for the divergence. The use of histograms makes it possible to compare the shots using different distances. The distances used were:

- Euclidean distance (L2): Distance that would be measured if we had 2D vectors represented in a plane and a ruler.

$$dL_2(h1,h2)=\sqrt{\sum_{i=1}^{n}(h_1(i)-h_2(i))^2}$$

- Bhatacharyya distance: This distance is usually applied to measure the similarity between data sets.

$$d_{Bhat}(h_1,h_2)=-\log\sum_{i=1}^{n}\sqrt{\frac{h_1(i)\cdot h_2(i)}{|h_1|\cdot|h_2|}}$$

After some experiments I decided that the best results were obtained when using 7 bins for horizontal motion and 5 for divergence (the example shown above), and results with L2 were slightly better than using Bhatacharyya distance.

Due to a lack of time it was not possible to integrate my work with the final system, so I only made some experiments using the Euclidean distance, but results were not so bad. What I did was a k-means algorithm trying different options: different number of bins in the histograms, only use horizontal motion, give less weight to the divergence... And I tried all that in a set of 30 videos distributed in 3 categories that k-means had to group correctly only knowing the expected number of clusters.

After running the algorithm only 4 shots were grouped wrongly, obtaining the following results:

Precision = 0,6763

Recall = 0,8182

F_1 = 0,7405

All those measures are explained in a previous article, although they are explained for image retrieval, they can be used in clustering if we analyse each possible pair instead of the documents retrieved.

As I said in the beginning of the article, I am currently writing my report and finishing some of the work that I could not do. I would like to compare histograms using other distances like Manhattan distance or Chi-square test, and run the k-means algorithm with all of them, to ensure that the best option is the Euclidean distance. Finally, it would be interesting to try another algorithm where the user does not have to give the number of clusters (the *k* of k-means), so no previous information is given.

Table 17 Article written in Bitsearch blog, published in July 2012

B. 5H+3D bins results

L2 Distance Horizontal Translation

	1	3	7	12	18	6	15	26	30
1		0,29	0,42	1,26	1,26	1,26	0,91	0,73	1
3			0,39	1,14	1,14	1,14	0,73	0,51	0,84
7				0,85	0,85	0,85	0,55	0,44	0,62
12					0	0	0,46	0,69	0,39
18						0	0,46	0,69	0,39
6							0,46	0,69	0,39
15								0,24	0,2
26									0,82
30									

Table 18 Euclidean distance of horizontal travelling histograms (fragment)

Bhattacharyya Distance Horizontal Translation

	1	3	7	12	18	6	15	26	30
1		0,02	0,04	0,53	0,53	0,53	0,16	0,1	0,25
3			0,08	0,5	0,5	0,5	0,1	0,04	0,19
7				0,2	0,2	0,2	0,1	0,09	0,14
12					0	0	0,1	0,18	0,09
18						0	0,1	0,18	0,09
6							0,1	0,18	0,09
15								0,009	0,04
26									0,07
30									

Table 19 Bhattacharyya distance of horizontal travelling histograms (fragment)

L2 Distance Divergence

	1	3	7	12	18	6	15	26	30
1		0,43	0,19	0,02	0,01	0,02	1,1	0,92	1,02
3			0,38	0,44	0,42	0,44	0,71	0,52	0,59
7				0,21	0,2	0,21	0,98	0,82	0,94
12					0,01	0	1,12	0,94	1,03
18						0,01	1,1	0,93	1,02
6							1,12	0,94	1,03
15								0,2	0,28
26									0,17
30									

Table 20 Euclidean distance of divergence histograms (fragment)

Bhattacharyya Distance Divergence									
	1	3	7	12	18	6	15	26	30
1		0,07	0,02	0,003	0,001	0,003	0,36	0,22	0,25
3			0,12	0,08	0,06	0,08	0,14	0,06	0,05
7				0,03	0,03	0,03	0,3	0,2	0,28
12					0,002	0	0,47	0,28	0,29
18						0,002	0,36	0,21	0,23
6							0,47	0,28	0,29
15								0,01	0,03
26									0,01
30									

Table 21 Bhattacharyya distance of divergence histograms (fragment)

C. Commented code

```
function [a,b,c,d,e,Fsc]=clustering5F(data)

GroundTruth=[1;3;2;3;2;2;2;3;1;3;3;3;3;3;2;2;2;2;2;2;2;2;1;2;1;1;3;1;1;3];
[row,col]=size(data); %Matrix size
mat=([data GroundTruth]); %Added as the last column
F=zeros(1,5);

for k=1:5

   index=randperm(30); %Random list of non-repeated values from 1 to 30

   for i=1:30
      imat(i,:)=mat(index(i),:); %The matrix rows are randomised
   end

   iGroundTruth=imat(:,col+1); %The new GroundTruth columns is extracted
   [y,c]=kMeansCluster(imat(:,[1:col]),3,1); %Kmean receives the
aleatorised matrix, 3 clusters and will use random centroids (1)
   F(k)=Fscore(y(:,col+1),iGroundTruth); %F1-Score calculation

end

a=F(1)
b=F(2)
c=F(3)
d=F(4)
e=F(5)
Fsc=(mean(F)) %Mean F1-Score is calculated and printed in command window,
single values are as well
```

Table 22 Clustering5F function code

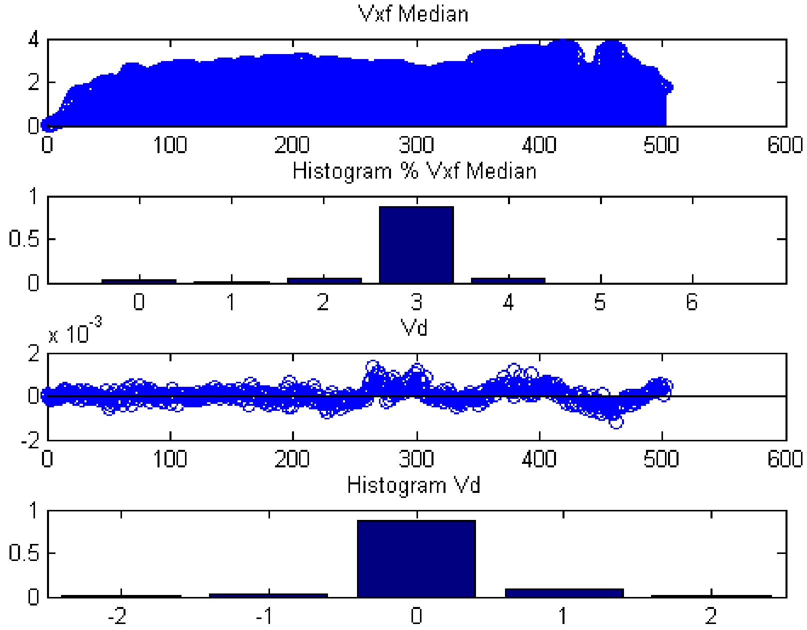
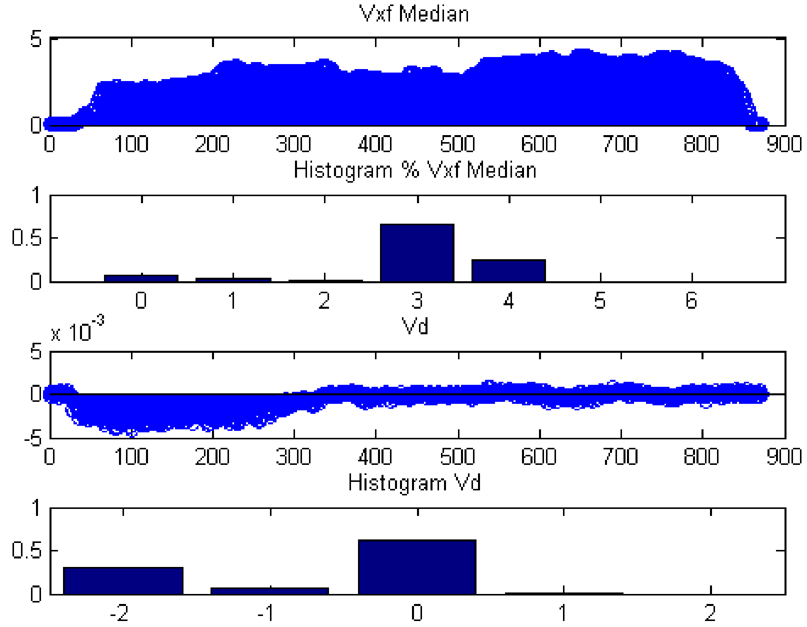
```
function F=Fscore(Results,GroundTruth)

TP=0; %True Positive
FN=0; %False Negative
FP=0; %False Positive
TN=0; %True Negative

for i=1:30
   for j=(i+1):30
      if(GroundTruth(i)==GroundTruth(j)&&Results(i)==Results(j))
         TP=TP+1;
      elseif(GroundTruth(i)==GroundTruth(j)&&Results(i)~=Results(j))
         FN=FN+1;
      elseif(GroundTruth(i)~=GroundTruth(j)&&Results(i)==Results(j))
         FP=FP+1;
      elseif(GroundTruth(i)~=GroundTruth(j)&&Results(i)~=Results(j))
         TN=TN+1;
      end
   end
end

P=TP/(TP+FP); %Precision
R=TP/(TP+FN); %Recall
F=2*P*R/(P+R); %
```

Table 23 F_1-Score function code

```
function d=DistMatrix(A,B)
% DISTMATRIX return distance matrix between points in A=[x1 y1 ... w1] and
in B=[x2 y2 ... w2]
% Copyright (c) 2005 by Kardi Teknomo,  http://people.revoledu.com/kardi/
% Numbers of rows (represent points) in A and B are not necessarily the
same.
% It can be use for distance-in-a-slice (Spacing) or distance-between-slice
(Headway),
% A and B must contain the same number of columns (represent variables of n
dimensions),
% first column is the X coordinates, second column is the Y coordinates,
and so on.
% The distance matrix is distance between points in A as rows
% and points in B as columns.
% example: Spacing= dist(A,A)
% Headway = dist(A,B), with hA ~= hB or hA=hB
%          A=[1 2 3; 4 5 6; 2 4 6; 1 2 3]; B=[4 5 1; 6 2 0]
%          dist(A,B)= [ 4.69   5.83;
%                       5.00   7.00;
%                       5.48   7.48;
%                       4.69   5.83]
%          dist(B,A)= [ 4.69   5.00     5.48    4.69;
%                       5.83   7.00     7.48    5.83]

[hA,wA]=size(A); %Elements
[hB,wB]=size(B); %Centroids
if wA ~= wB, error(' second dimension of A and B must be the same'); end

   d=2; %L1-->1, L2-->2, Bhat-->3

   %L1
   if d==1
   for k=1:wA
      C{k}= repmat(A(:,k),1,hB); %Matrix with replicated columns
      D{k}= repmat(B(:,k),1,hA);
   end
   S=zeros(hA,hB); %Empty matrix to be filled with distances
   for k=1:wA
      S=S+abs(C{k}-D{k}'); %L1  calculation
   end
   d=S;
 end

%L2
if d==2
   for k=1:wA
      C{k}= repmat(A(:,k),1,hB);
      D{k}= repmat(B(:,k),1,hA);
   end
   S=zeros(hA,hB);
   for k=1:wA
      S=S+(C{k}-D{k}').^2; %L2 calculation
   end
   d=sqrt(S);
end

%Bhattacharyya
if d==3
   for k=1:wA
      C{k}= repmat(A(:,k),1,hB);
      D{k}= repmat(B(:,k),1,hA);
   end
   S=zeros(hA,hB);
   for k=1:wA
      S=S+(sqrt(C{k}.*D{k}')); %Numerator
   end
   for i=1:hA
      for j=1:hB
         S(i,j)=S(i,j)/sqrt(norm(A(i,:),1)*norm(B(j,:),1)); %Denomin.
      end
   end
   d=-log10(S);
end
```

Table 24 DistMatrix function code

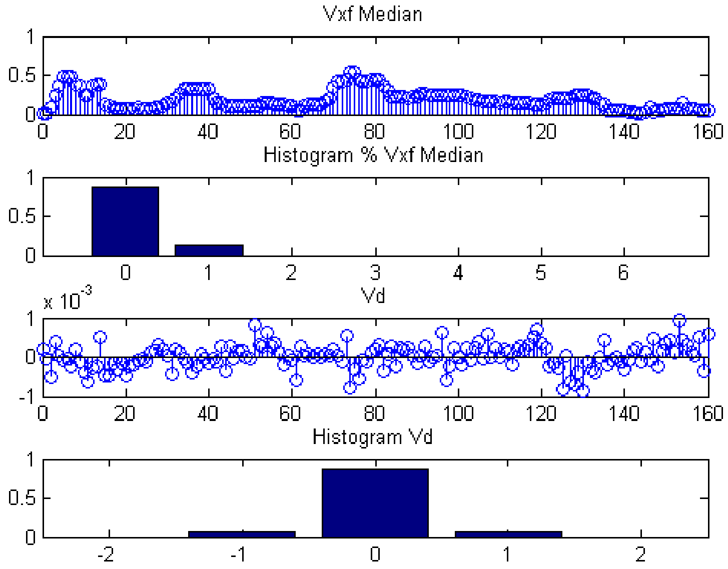
D. 7H+5D bins histograms

<table>
<tr><td colspan="2">Clip 1: Travelling static scenario</td></tr>
<tr><td>1st image</td><td rowspan="4">
</td></tr>
<tr><td> </td></tr>
<tr><td>503</td></tr>
<tr><td> </td></tr>
<tr><td colspan="2">Clip 3: Travelling static scenario</td></tr>
<tr><td>1st image</td><td rowspan="4">
</td></tr>
<tr><td> </td></tr>
<tr><td>872</td></tr>
<tr><td> </td></tr>
</table>

Clip 6: Bust

1st image

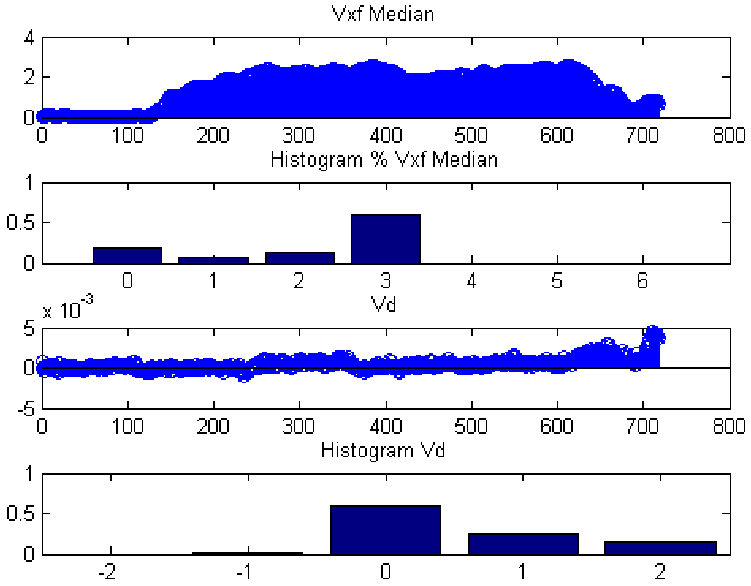
161



Clip 7: Travelling static scenario

1st image

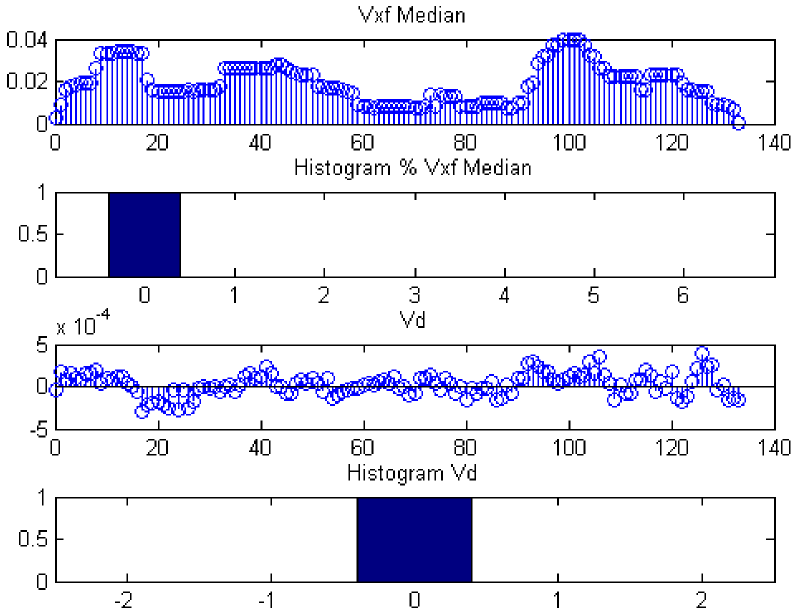
719



Clip 12: Bust

1st image

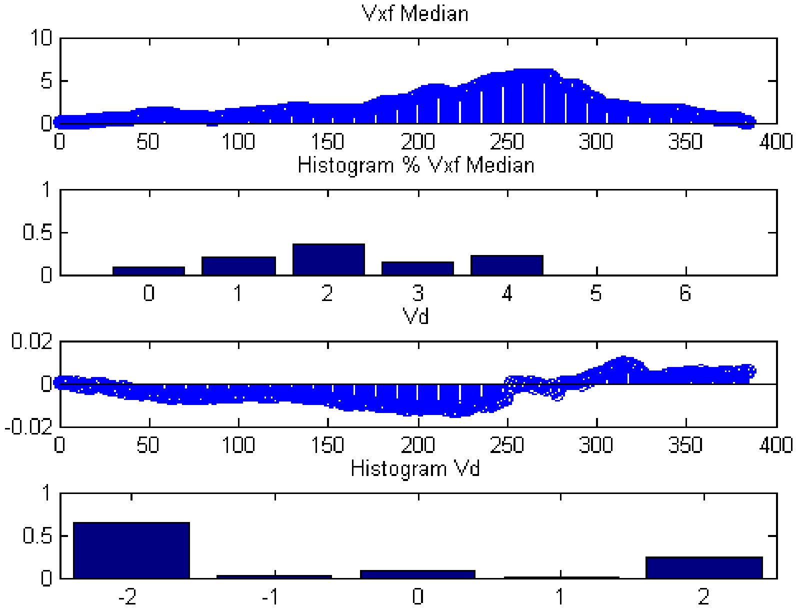
134



Clip 15: People walking (Tracking)

1st image

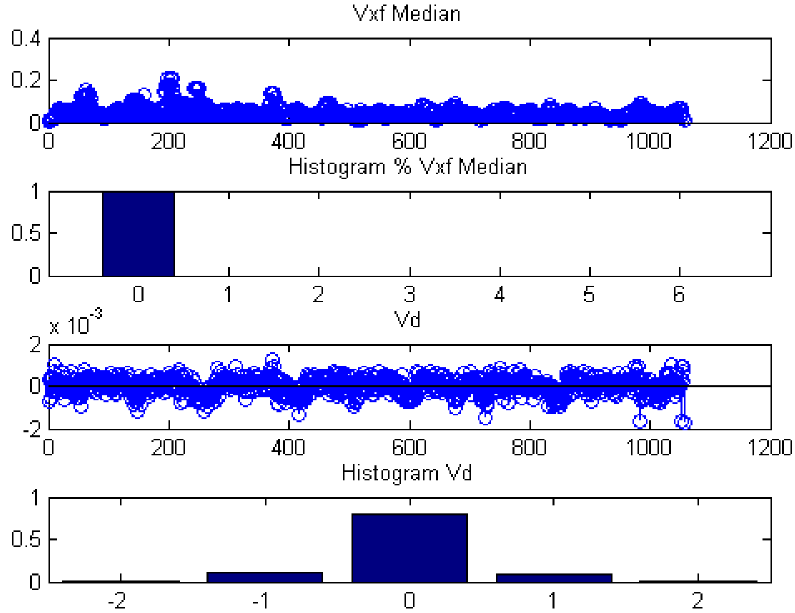
386



Clip 18: Bust

1^{st} image

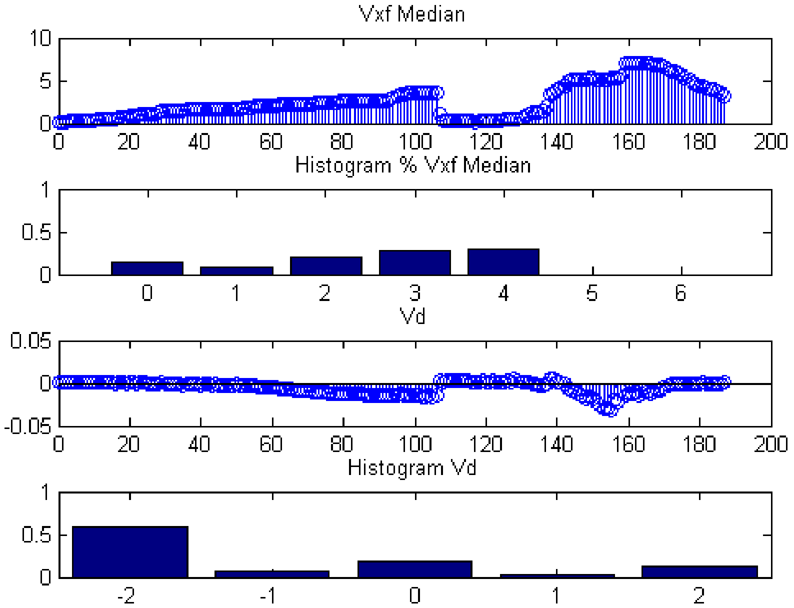
1058



Clip 26: Vehicle (Tracking)

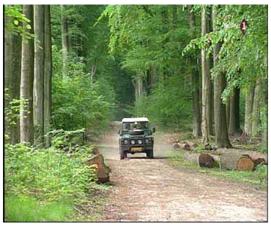
1^{st} image

188

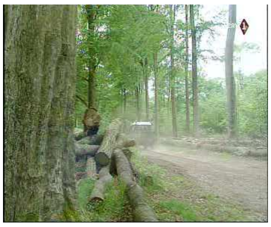


Clip 30: Vehicle (Tracking)

1st image



251



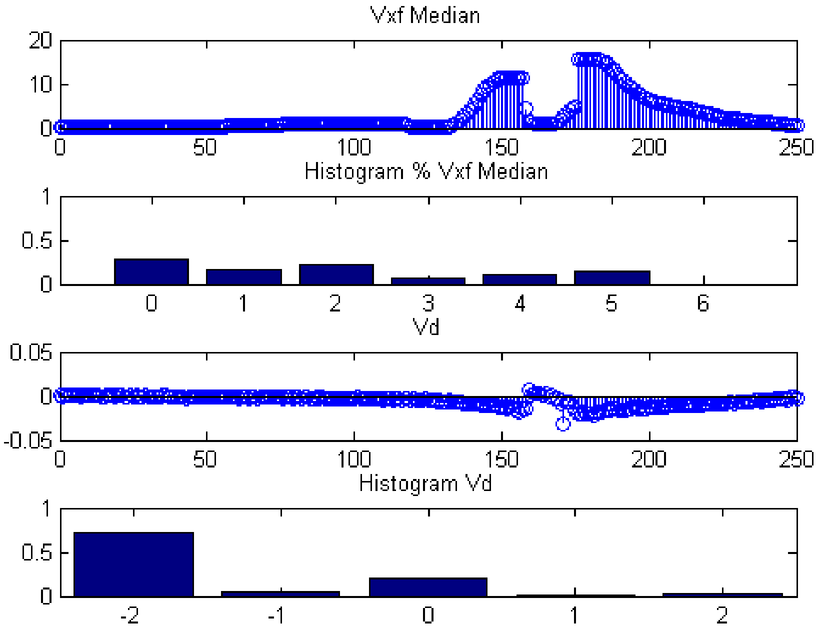


Fig.40 Histograms of the clips analysed in section 5.3

E. F$_1$-Score results

Bins						Mean
5+0	0,7133	0,5438	0,5438	0,5935	0,5417	0,5872
5+3	0,6373	0,4631	0,698	0,5127	0,4631	0,5548
5+5	0,5208	0,6376	0,4631	0,6376	0,6376	0,5793
6+0	0,6429	0,6078	0,6429	0,6845	0,6458	0,6448
6+3	0,5603	0,637	0,6458	0,6376	0,4939	0,5949
6+5	0,7483	0,6516	0,5015	0,5106	0,5286	0,5881
7+0	0,5385	0,5248	0,6181	0,5248	0,5571	0,5527
7+3	0,4955	0,5044	0,6376	0,4825	0,5532	0,5526
7+5	0,5385	0,6443	0,5528	0,443	0,6376	0,5632

Table 25 Manhattan distance F$_1$-score results

Bins						Mean
5+0	0,5466	0,6926	0,5438	0,5935	0,5417	0,5836
5+3	0,6376	0,5385	0,6376	0,6867	0,4863	0,5973
5+5	0,5397	0,5594	0,5371	0,6867	0,6867	0,6019
6+0	0,6915	0,6429	0,6164	0,6429	0,7048	0,6597
6+3	0,6376	0,6376	0,6369	0,6458	0,6458	0,6807
6+5	0,5774	0,5774	0,6084	0,7071	0,6059	0,6152
7+0	0,6	0,6294	0,5625	0,5625	0,6	0,5909
7+3	0,5248	0,5831	0,5724	0,6376	0,6376	0,5911
7+5	0,7009	0,7009	0,6316	0,5532	0,6316	0,6436

Table 26 Euclidean distance F_1-score results

Bins						Mean
5+0	0,5366	0,5015	0,5333	0,5245	0,5686	0,5329
5+3	0,5015	0,6515	0,5	0,6071	0,6915	0,5903
5+5	0,6266	0,5592	0,6738	0,4631	0,5122	0,567
6+0	0,5592	0,4583	0,6845	0,5714	0,5549	0,5657
6+3	0,6071	0,6149	0,6738	0,7055	0,4658	0,6134
6+5	0,7533	0,6376	0,4365	0,5745	0,4658	0,5735
7+0	0,5714	0,6181	0,5592	0,5915	0,5592	0,5799
7+3	0,4955	0,4658	0,6458	0,5816	0,6357	0,5649
7+5	0,5695	0,5915	0,5583	0,6458	0,5248	0,578

Table 27 Bhattacharyya distance F_1-score results