SEMI-SUPERVISED EXTRACTION OF AUDIO-VISUAL SOURCES

Master Project

EPFL

PATRICIA CALATAYUD MARTINEZ

Assitant:
ANNA LLAGOSTERA CASANOVAS
Supervisor:
Prof. PIERRE VANDERGHEYNST

Lausanne, March 31, 2010

Contents

Chapter 1

Introduction

1.1 Abstract

This report presents a semi-supervised method to jointly extract audio-visual sources from a scene. It consist of applying a supervised method to segment the video signal followed by an automatic process to properly separate the audio track.

This approach starts with the user interaction to select the audio-visual target we want to cut. This labels the problem into two parts. One is the foreground, the object we aim to extract. The background is the remaining content which sets the other part.

The segmentation method will be performed over the video signal to split its visual content into foreground and background. It is based on describing the video information as a volume. It is composed of pixels and relationships among them which set the features of the problem. This 3D structure can be interpreted as a graph. Therefore the segmentation is achieved by applying a Graph Cuts algorithm which provides the suitable results.

In this work the audio separation process is an automatic task. Due to this we need to extract prior information from the audio-visual data available. Therefore a link between audio and video channels is defined by an audio-visual motion map. This is a video sequence which provides the amount of synchronous motion with the audio track. An audio-based video diffusion method have been developed to obtain this kind of signal.

Applying the video segmentation labelling to the audio-visual motion map we obtain the distribution of the amount motion into foreground and background. According to the range of these values it is possible to asses when the corresponding audio is on/off. At this point, audio samples of sources can be extract when the audio activity is due to only one. This is the prior information to

learn audio models by means of spectral GMM estimation to assess the audio signals.

Finally the separation method is applied. It consist of looking for the more suitable couple of states given the mixture spectrum of the total audio track. Therefore we achieve the audio separation goal.

In this work we will study in detail all the stages towards the final aim of our method, analyzing the performances of our approach.

1.2 Problem statement

Audio-visual sequences are signals which jointly present useful information to process for different purposes. For instance, it is very attractive to think about working with musical video clips where we can carry out artistic changes with its audio-visual content.

From a point of view of signal processing these signals are commonly separated to interpret and modify the data they provide. It is more interesting to try to handle audio and video channels together in order to profit the maximum information available. In order to do this a relationship to relate them must be found. This can be intuitively perceived by the movement related to a source of sound in a scene. Assessing the synchrony between these kind of events, the audio-visual connection is discovered.

Thereby the challenge we decided to face in this work is performing a kind of audio-visual processing to exploit the correspondence between audio and video modalities. We provide an approach to handle an audio-visual sequence starting from the video segmentation of its visual content. Furthermore video information will be processed according to its audio content and the link between them. The extracted information from both methods is jointly interpreted to finally process the audio signal in order to achieve its separation.

Video segmentation face the problem of separating a moving foreground from a not trivial moving background for practical purposes as pasting the result into a new context. The foreground will be the selected object by the user into the video. The background is the remaining part of the which can be composed of several elements. Foreground and background can change its definition, but we will assume for the rest of our work, our target will be the object. Our aim will be to achieve an efficient video segmentation, this means with a user interaction as minimal as possible. As we can read in [15] several approaches have focused on solving this problem. Among them we will introduce briefly , the most meaningful of them, in the framework of our work.
Magic Wand is based on user-specified points or regions by means of an interface. The procedure try to find out where they match according to color statistics of the image separation. Finding a suitable result is sometimes difficult mainly when colors of foreground and background overlap.
Intelligent Scissors shows the minimum cost path for a fixed "minimum cost

contour" of the object depicted by the user with the mouse. The main problem appears if several minimal paths exist when this is applied in textured or untextured regions.
Bayes matting solves the problem with a "trimap" $\mathcal{T} = \mathcal{T}_B, \mathcal{T}_U, \mathcal{T}_F$ for foreground, background and unknown regions given by an user interaction. This information models colour distributions probabilistically to achieve high quality mattes but only when unknown region is not too large and color distributions are separated enough.
Poission matting works with trimaps as Bayes matting. It computes an alpha matte by solving the second order equation with Dirichlet boundary conditions and given assumptions over the gradient information. This is mentioned in [2] too.
Graph Cut [8] is the method we have to highlight because is the basis of our work. It take advantage of trimaps and probabilistic color models to obtain satisfactory segmentations. It is powerful to extract the target even when foreground and background have similar color distributions.
Level sets is a tool to minimize almost any energy function. It works with a partial differential equation to compute a local minimum depending on initialization. When Graph Cuts is able to do it exactly, that method will be most suitable than this one.
Grab Cut algorithm introduces iterative estimation and incomplete labelling contributions to the former Graph Cut method. This simplifies the user interaction considerably but can propagate segmenting errors hurting the segmentation results as it is explained in [2]
Between these and other graph-based techniques which aim to contribute new features to this framework as it is analyzed in [5], we will apply Graph Cuts first introduced in [8] and recently further developed in [6], to obtain a 3D segmentation of a video signal as we stated before. Graph Cut is a method which optimize a discrete energy function which combines region and boundary properties of the selected regions. Furthermore it allows to include topological constraints that naturally fit into our global optimization framework. This is pointed in [5] It addresses the foreground/background interactive separation via max-flow/min-cut formulation. We will use this system as the better tool to process N-D signals and color images and videos. We will focus on this in the corresponding Chapter.

The Audio processing part of our method exploits the audio-visual synchrony between audio and video modalities. This is the key point of our audio sources separation. For the audio modality we aim to do the same as for the video modality. This means we want to separate the foreground audio source from the background audio source. However we have to notice that one of the audio targets can be composed of several audio tracks as it happens with the visual labels. In the development of our work we will suppose this assumption is known, and we do not explicitly refer to this concept, but when necessary.
Audio-visual analysis is based on the fact that visual information strongly contributes to the interpretation of acoustic signals. This technique is enhancing its importance because it is at the basis of a broad range of applications.
The main attempts to complete audio-visual source extraction using a video signal and the corresponding soundtrack are explained in [14]. Among other works, they revise the two main approaches developed in this field to obtain a

complete audio-visual source separation. The attempt developed in [3] and the study performed in [18] are based on assessing the correlation between audio and video structures. The first one works with a relation of frequency formants between audio and video elementary events defined as onesets. The second one uses trajectories of "interest " points to establish the relation between audio and video. The method shown in [14] presents a novel non-linear diffusion process as the first stage to the total audio separation. This procedure will give us the periods of audio activity/inactivity for the problem sources computed from an audio-visual motion map. This information will let us to extract training audio samples for both when they are active alone. With the obtained data we can estimate audio models for foreground and background by means of Gaussian Mixture Models (GMM) as it is discussed in [14]. Finally we will asses the audio track with the GMM models and we will get the desired audio separation.

Once we achieve the total result we can process the separated target to composed new video signals adding the desired audio-visual foreground into a new background.

1.3 Overview

The algorithm overview of our system will let us to present how we have addressed the structure of our work.
The main targets of our method are audio-visual sequences signals in which two or more sound sources as musicians are mixed. One of them sets the foreground. The rest will be labelled as background.
The challenge would be to separate both, in an audio-visual sense. We will apply the corresponding algorithms to both modalities following the necessary assumptions to achieve the desired results. For the suitable audio extraction, there two requirements that the features of our input signal have to respect.
First we assume that there is at least one audio-visual source in the background. It will allow to perform the detection of audio activity of sources.
Second assumption is audio-visual sources for foreground and background alternate between on/off states. This condition will let us to detect periods during which audio-visual sources are active alone and periods during which they are mixed. Thereby we will be able to learn audio models which lead to the audio separation of our targets. The method has been performed in several steps which can be mainly classified as video and audio stages.
The flow diagram of our work is illustrated in Figure 1.1.

In a first step towards our aim, the video segmentation procedure is applied to the input video signal. At this starting point, it is necessary the user interaction in order to provide the prior information of the sources. The user must draw the basic seeds or rough scribbles over the visual content. It will label the regions of interest as foreground and background for the rest of the process. Because of this reason our work is characterized as a semi-supervised technique.
The segmentation acts defining the video volume as a 3D structure. This can be interpreted as a graph composed of pixels and the link among them set its

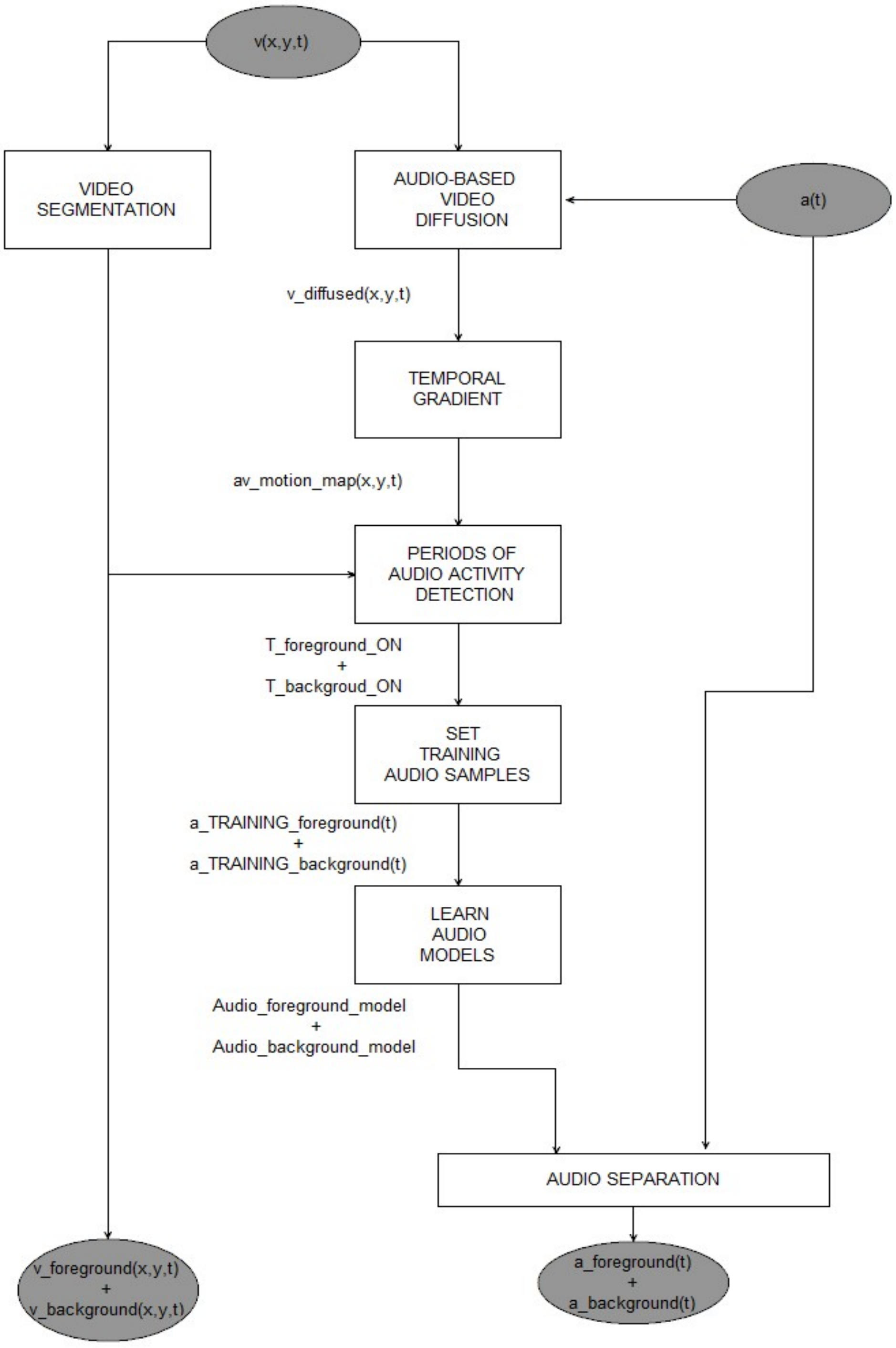

v(x,y,t)
VIDEO SEGMENTATION
AUDIO-BASED VIDEO DIFFUSION
a(t)
v_diffused(x,y,t)
TEMPORAL GRADIENT
av_motion_map(x,y,t)
PERIODS OF AUDIO ACTIVITY DETECTION
T_foreground_ON + T_backgroud_ON
SET TRAINING AUDIO SAMPLES
a_TRAINING_foreground(t) + a_TRAINING_background(t)
LEARN AUDIO MODELS
Audio_foreground_model + Audio_background_model
AUDIO SEPARATION
v_foreground(x,y,t) + v_background(x,y,t)
a_foreground(t) + a_background(t)

Figure 1.1: Semi-supervised extraction of audio-visual sources Flow Diagram

features. An energy function is assigned to the graph connections which is composed of regional and boundary information for the pixels. Thus, the video segmentation is described by a Graph Cut technique which solves the problem by minimizing this energy function. Our approach presents two new features recently estimated by comparing with former methods.
The first one is a novel formulation of the regional term of the energy function. This will let us to outperform the results obtained by previous studies as the ones developed in [8].
The second one consists on splitting the video-signal in 3D Block of Frames (BOF). This feature will allow us to work faster and easier with a huge amount of pixels. It means we will be able to process sequences with the suitable length for our purposes. The idea of the procedure is to segment one BOF and extend the result to the following one. This will be achieved sharing one frame between each neighboring BOF. For the first BOF, the foreground and background prior information is manually given by the user as we introduced before. However for the remaining BOF the method will profit the resulting segmentation of the shared frame with the previous block. According to this, the labelling and the evaluation of the initial data of our regions is obtained automatically. Furthermore, this way of processing the video-signal gives us the chance to improve the resulting segmentation for each BOF. It will be done by the user supervision again, adding more labelling information, and applying segmentation again, before its propagation. Despite this we will try to achieve the ideal result reducing as much as possible the user effort.
We will study the method in Chapter 2.

At the same time, and given the video signal and the audio track as inputs, we can deal with the procedure to link the audio and video channels. Audio-based video diffusion is a procedure to find its correspondence by estimating the synchrony between them. This will be performed as a boundary problem controlled by a scalar-valued diffusion coefficient which we have to solve. Thus, movements non correlated to the sound are eroded whereas motion with an important coherence with the audio track is preserved.
Therefore we obtain a diffused video which will be applied to a temporal gradient procedure. As a result we will have a useful audio-visual motion map, the motion interpretation of the audio track .
Using this information and the video segmentation we can face the procedure of audio activity detection. The cutting mask for foreground and background is applied over the audio-visual motion map. We compute the amount of motion which corresponds to each of the sources at each interval. According to the relation between the range of motion values and acoustic events, the periods of audio activity are finally extracted.
This will be analyzed in detail in Chapter 3.

Next,the result of the audio-activity detection will be profited to provide the audio samples where sources are playing alone from the original audio signal . These values will be the training points for our audio GMM estimation. With this information will build audio models. Assessing the audio track matching for each source, finally our remaining objective of separating audio targets is achieve.
This challenging process will be explained in Chapter 4.

Explained the methodology to solve our problem, we will focus on the Graphical User Interface which we have designed as the essential tool to manage this semi-supervised process. It integrates all the procedures in order to provide the jointly extraction of audio-visual sources.
It will be addressed in Chapter 5.

Chapter 6 summarizes the practical work carried out in order to test and assess our process.

Finally Chapter 7 will presents the conclusions and future scenario of the whole work.

Chapter 2

Semi-supervised extraction of the visual content of the source

2.1 Introduction

The first step of our approach consists of handling a video signal by applying a segmentation procedure. Our goal is to separate the visual content of each video frame into two regions. One region is the foreground which identifies the visual component of the audio-visual element we aim to extract. Only one object will be included into this region as we discussed in the introduction, to achieve our purposes. Considering the kind of sequences we work with, the foreground would be an audio-visual source as a musician, a singer or a simple source of sound as a speaker. The other region is the background composed of the remaining visual content. This means the background will contain the rest of the audio-visual sources of the video signal. They will be the same elements as we have pointed out for foreground. Therefore in order to perform our video segmentation procedure, prior information about the audio-visual targets is required and user interaction is essential. User has to label the visual content of the video as foreground and background as the starting point of our process.

Once the objective of video segmentation of a video signal is explained we have to find out how to solve our problem. A video signal can be represented as a volume composed of frames along the time. Numerically this can be built as graph of nodes and connections. Pixels will be the nodes and the links between them set the connections of the graph. An energy function will be assessed according to relationships between pixels and this will be assigned into the graph. Then the video segmentation will be carried out as a procedure over the corresponding graph in order to achieve the suitable cut. Thereby our aims will be to find out the appropriate methods to built the graph, to perform its cut and to achieve the most accurate results taking advantage of the features of the procedure provided. Furthermore we try to solve it as efficiently as possible according to the user supervision.

2.2 Formulation of the problem: Graph Cuts

As we introduced in the previous Section, in the first step of our method we work with a video signal in order to extract a foreground from a remaining background selected by the user.
When we try to face our objective of video segmentation we find a 3D problem which will be given by the video frames segmentation over the time. We work with color intensities, space and time. This 3D structure is transformed into a graph.

A graph $\mathcal{G}$ is defined as a pair $\{\mathcal{V}, \varepsilon\}$, where $\mathcal{V}$ is a set of vertices, and ε is a set of edges between the vertices $\varepsilon = \{(u, v)|u, v \in \mathcal{V}\}$.
A weighted graph is a graph having a weight associated to each edge.
In our framework, vertices are pixels and the relationships among them will build the weighted edges of our graph. Some algorithms as the one we use to solve the problem, require all weights to be integer and positive.
Our graph will be represented by an adjacency-matrix, an vxv matrix, where v is the number of vertices and the entry at {i,j} is the weight of the edge if there is a relation from vertex i to vertex j; otherwise the entry is 0.
Also our graph is undirected, we have the same entry in {i,j} and {j,i} so we can use an upper triangular matrix to build the adjacency-matrix.

Using this formulation for our problem, the segmentation of one region from a video signal is translated into a cut over its graph as we can read in [**?**]. It will consist of finding a cutting path such as it minimizes certain function implemented in the weighted edges of the graph. Therefore the vertices and the edges will be separated into two mutually exclusive sets: S whose points are called sources and T whose points are called sinks. These labelled and separated regions form the cut.
In weighted graphs, the weight, or the capacity, of a cut is the sum of weights of the edges that have exactly one endpoint in each of the two sets.

Then, the problem of finding the minimum weighted cut in a graph is called Minimum Cut Problem. Furthermore as we can learn in [**?**] too, finding the minimum cut over a graph is equivalent to find a maximum flow through it. If s is the source and t the sink in a graph $\mathcal{G}$, the Maximum Flow - Minimum Cut theorem says:

$$Max - Flow(s, t) = Min - Cut(s, t)$$

This means,our problem will be to find out the maximum amount of flow capable to pass from a source into a sink via a network following the model of flows in a graph as a water conduit in pipe networks as explained in [**?**]. The theory about Water flow in networks is defined by:

“ Suppose we have a water source, a sink and a network of pipes relating them. We accept that the source can give (and the sink can receive) an unlimited amount of water. Supposing the unlimited capacity for the source and the sink, the problem is only constrained by the capacity of the network. The capacity of each pipe is proportional to its diameter. In this network, there will be obstructions in some pipes. To pass from to the source to the sink, water must absolutely take one of these obstructed pipes. In an intuitive case, if all the pipes are full, the flow is equal to the sum of their capacities. In particularly, the obstruction corresponds to a set of pipes, which separates the source from the sink, and the sum of its capacities is equal to the maximum flow capable to pass from the source to the sink. Once the flow is maximal, we are in the situation where all the pipes of obstruction are full (saturated).
In other words, the theorem states that the maximum flow in a network is dictated by its bottleneck.
Between any two nodes, the quantity of material flowing from one to the other cannot be greater than the weakest set of links somewhere between the two nodes.”

Then our segmentation has been formulated as the problem of finding a minimum cut - maximum flow through our video graph. Thus, we have to find out how to face this question. The Push-Relabel algorithm will be a suitable method to solve our problem [**?**]:

“ Suppose that the graph can be represented as follow: the source s is on the top and the sink t is at the bottom.
For each vertex we can define its height. Assigning to the source the maximum height $\mathcal{V}$ and setting the minimum height 0 to the sink, each vertex will have a height given by:

$$Height(s) = V, Height(t) = 0, \phi(u,v) > 0 \Longrightarrow Height(u) > Height(v)$$

.
We send water from the source down to the sink.
In each vertex, water in excess will flow to the vertices at smaller height, constrained with the capacity of the edge relating the two vertices.
After each operation, water will be in a lower vertex, and the excess is reduced in the upper vertex. We track water until arriving to the sink t. The algorithm stops when there is no excess in any vertex.
In other words, the heights of nodes (except s and t) are adjusted, and flow is sent between vertices, until all possible flow has reached t.
Then we continue increasing the height of internal nodes until all the flow that went into the network, but did not reach t, has flowed back into s. Let $\varepsilon(u) = P(V,u), \varepsilon(u)$ is called the excess of vertex u, which is the difference between the in and out flows of the vertex u.
Now, the total amount which can flow into a vertex can be bigger

than the flow out of the vertex (the principle of flow conservation is no longer respected). This is the notion of pre-flow is introduced by (Karzanov, 1974). which clarifies the excess concept.
A vertex u which belongs $\mathcal{V} - \{s, t\}$ is called overflowed if $\varepsilon(u) > 0$.
The algorithm overview of the general implementation of the Push-Relabel algorithm is shown in [**?**], as well as the Push and the Relabel operations defined in (Goldberg and Tarjan, 1988)."

The Push-Relabel algorithm is one of the most efficient algorithms to compute a maximum flow. The general algorithm complexity is O(v2e) [9], where v is the number of vertices in $\mathcal{V}$ and e is the number of edges in $\mathcal{E}$. In our work, we use an implementation of Push-Relabel algorithm proposed by [11], which is included in Boost Graph Library1 [17].

2.3 Features: the energy function

We have established the basis of our video segmentation problem as minimization of a discrete function assigned to a weighted and undirected graph. It can be directly computed with an exact graph-based optimization procedure. We apply the Graph Cuts method, which will solve the problem applying the Push Relabel Algorithm to perform a Maximum flow - Minimum Cut. Now we have to set the features to our problem in order to deal with its resolution.

The segmentation can be formulated as a labelling problem [7].
The objective is to find such a labelling f that assigns to each pixel $p \in \mathcal{P}$ a label $f_p \in \mathcal{L}$, where f is both piecewise smooth and consistent with the observed data. Labelling is a map from $\mathcal{P}$ into $\mathcal{L}$:

$$f : \mathcal{P} \longrightarrow \mathcal{L} : s_P \longrightarrow f_p = f(s_P) = l_i$$

In our work we perform a hard segmentation thus our labels are binary $f_p \in \{0, 1\}$ where:
"1":this is the value which represents the foreground label.
"0":this is the value associated to the background label.
In our framework we work with $z = (z_1, .., z_p, z_P)$ as the set of pixels in the RGB color space that compose a 3D video fragment. As we stated before, we fix our labels as foreground (our selected visual object) and background (the remaining visual content of the video file). Then the graph which implements the labelling problem consists of $\mathcal{G} =< \mathcal{V}, \varepsilon >$ It is composed of a set of vertices $\mathcal{V}$ corresponding to our image pixels. In addition two of them are labelled as special nodes, the one which identifies the foreground $\mathcal{F}$ and the one which identifies the background $\mathcal{B}$. The set of graph edges will be given by the relationships between neighboring pixels ($\mathcal{N}$-links) as well as by the relationships between each pixel and the labelling nodes ($\mathcal{T}$-links). We will set four spatial neighboring pixels and two temporal neighboring pixels for each one. We will consider this situation changes at graph borders and corners and removing non-existent links the structure will be adapt. Finally the features of the edges between the pixels

will build the undirected weights of the graph. They translate the meaning of the segmentation process into the labelling problem of the graph.

Therefore our aim will be to perform a labelling f that minimizes a function defined by the energy relationships among the pixels of our graph:

$$E(f) = E_{boundary}(f) + E_{regional}(f) \tag{2.1}$$

This is the "Gibbs" energy which will be the function associated to the edges of our graph.
This function E is defined so that its minimum should correspond to a right segmentation, in the sense that it is guided both by the observed tendency to solidity of objects and by foreground and background color distributions.

The first term $E_{boundary}(f)$ measures the similarity in terms of intensity between neighboring pixels. This is the characteristic weight associated to N-Links. This energy encourages coherence in regions of similar intensity level. This quantity will give us information about the shapes into the image. We could state that a change of object can be translated into an important difference of intensity between neighboring pixels or edges.
We define $E_{boundary}$ as:

$$E_{boundary}(f) = \Sigma_{p,q\epsilon\mathcal{N}} B_{p,q} \delta_{f_p \neq f_p} \tag{2.2}$$

The boundary term is defined by:

$$B_{p,q} = \beta \frac{1}{dist(p,q)} exp(\frac{-\parallel (z_p - z_q) \parallel^2}{2\sigma^2}) \tag{2.3}$$

Where:
$\beta = 100$: this is a constant which we use to set the range of boundary weight values; we have estimated an appropriate range between [0,100].
$dist(p,q) = 1$: this is the Euclidean distance between pixels;
It is set to 1 in this work because as we said we work with neighboring pixels so the distance between them is 1.
$\parallel (z_p - z_q) \parallel$: this is the norm 2 for intensity pixels in the RGB color space which are handled as 3D points.
$\sigma^2 = <\parallel (z_p - z_q) \parallel^2>$: this is the expectation operator for the intensity of all the pixels we are working with.
This boundary function assigns a big penalization to pixels with similar intensities, i.e. $\parallel (z_p - z_q) \parallel < \sigma$. However, it sets the penalization to low values when the intensities for the pixels are very different, i.e. $\parallel (z_p - z_q) \parallel > \sigma$.

In this framework of boundary relations, it remains to specify the meaning of the T-links. These are the connections from pixels to labelled regions. We set one pixel into each region as its main point and we link the other ones to them with a harder weight. These are the called hard constraints. We assume that $\mathcal{F}$ and $\mathcal{B}$ denote the subsets of pixels a priori known to be a part of "foreground" and "background" correspondingly. As we explained before these information is provided by the user interaction to define the audio-visual regions which aim

to separate. Naturally, the subsets $\mathcal{F} \subset \mathcal{P}$ and $\mathcal{B} \subset \mathcal{P}$ are such that $\mathcal{F} \cap \mathcal{B} = 0$
So, we can defined the hard constraints by:

$$\forall p \in \mathcal{F} : f_p = \text{“}\textit{foreground}'' \tag{2.4}$$

$$\forall p \in \mathcal{B} : f_p = \text{“}\textit{background}'' \tag{2.5}$$

This means, when hard constraints are provided, it is known a priori, they belong to foreground or background unquestionably. In order to link hard constraints to the main pixels of labelled regions we assign a weight to their edges defined by:

$$\varepsilon\{p, \mathcal{F} \mid \mathcal{B}\} = \mathcal{H} = 601 \tag{2.6}$$

We have tested the better choice to link hard constraints is 601 for our purposes. Ability to incorporate hard constraints is one of the most interesting features of our segmentation method.
There is a lot of flexibility in how these hard constraints can be used to adjust the algorithm for different tasks.
The hard constraints can be set either manually or automatically. Manually as user does to assign the prior information of the labelling problem and even to improve the segmentation results as we will explain. Automatically to profit the cutting mask to propagate the segmentation as will see in next steps of our process.

As we can learn in the literature related with Graph Cuts, the segmentation provided over our graph applying Maximum-flow/Minimum-Cut algorithm can be written as: $S = R + F$
S is the graph matrix we aim to cut with its features described as nodes and links.
F is the flow matrix with the values of used capacities into graph edges to compute the minimum cut.
R is the residual matrix with the values of unused capacities of the graph edges which provides the data to obtain the cut applying an algorithm over of finding minimum paths.
We can notice that re-writing the obtained information we can define the matrix R as: $R = S - F$
When we apply a re-segmentation over the current cut, this means to update the data structure for the graph. Analyzing the arithmetical relationship between matrix this can be extended to the residual matrix for the cut:
$R_{updated} = S_{updated} - F_{optimal}$
Thus we can notice the improving cut can be performed without recomputing the whole solution from scratch. The only requirement is assigning the proper conversions to entries of matrix $R_{updated}$ to respect the assumptions of weights in our graph [13]. Then we can proceed to solve the re-segmentation problem applying the algorithm of finding the optimal path over this matrix we have found.
This new approach of the re-segmentation of a graph cut involves a great reduction for the timing requirements of applying a cut. Thus this is feature would be interesting to develop as an improvement over the segmentation task in order to overcome the limitations related to time.

The second term $E_{regional}(f)$ evaluates how the color information of one pixel fits into the color models of our segmentation regions. This value will provide information about the probability of one pixel of belonging to each part of our binary labelling.

$$E_{regional}(f) = \lambda \Sigma R_p(f)) \forall p \tag{2.7}$$

The constant λ can control the relative importance of the energy terms; bigger values of λ give more importance to this feature. In our work we have fixed $\lambda = 0.05$ because we have estimated this is the proper value to rely on the boundary term and avoid to make segmentation worse when color information of foreground and background is similar.
The regional term is defined by:

$$R_p(f = \mathcal{F}) = \gamma f(ln(Pr(z_p|\Lambda^{\mathcal{F}}))) \tag{2.8}$$

$$R_p(f = \mathcal{B}) = \gamma f(ln(Pr(z_p|\Lambda^{\mathcal{B}}))) \tag{2.9}$$

Where:
$Pr(z_p|\Lambda^{\mathcal{F}})$: this is the conditioned probability for a pixel p to belong to the foreground/background given their color models.
$ln(Pr(z_p|\Lambda^{\mathcal{F}}))$: this is the Neperian logarithm of the computed probabilities.
$f(ln(Pr(z_p|\Lambda^{\mathcal{F}})))$: this is a function which acts over the mentioned logarithm to change our working range from]-∞,0] to [0,1],where:
"0":represents the lowest probability for the color of pixel z_p of fitting into the considered color model $\Lambda^{\mathcal{F}}$ or $\Lambda^{\mathcal{B}}$.
"1":represents the highest probability for the color of pixel z_p of matching the given color model $\Lambda^{\mathcal{F}}$ or $\Lambda^{\mathcal{B}}$.
$\gamma = 100$: this is a constant which acts over the range [0,1] of the regional term in order to get the suitable weight values; we choose 100 as we did for the boundary term; however we have to notice the range will be adjusted to [0,5] due to the effect of the λ parameter.

At this point we will have to focus on the method that we apply to model the color distributions of our labelling problem. It will be assessed by means of Gaussian Mixture Models.
We take advantage of Gaussian Mixture Models (GMMs) because we work on RGB colour space and so it is impractical to construct adequate colour space histograms [16] and GMMs are more robusts. Consequently, we use the contrast-sensitive GMMs to generate models for two layers color images (foreground and background) in order to get the simplest probabilistic model for interactive segmentation as it is discussed in [4].
The underlying assumption is that the colors of the pixels are generated by a mixture of Gaussians. Each GMM, one for the foreground and other for the foreground, is taken to be a full-covariance Gaussian mixture with K components (typically K = 5). Considering the conditioned probability for a pixel p to belong to the foregroung/backgroung we can define:

$$p(z_p \mid \Lambda^i) = \Sigma_k w_k^i K(\mu_k^i, \Sigma_k^i) \tag{2.10}$$

Learning a Gaussian mixture model is, in essence, an unsupervised clustering

edge	weight	for
$\{p, q\}$	$B_{p,q}$	$p, q \in N$
$\{p, F\}$	$\lambda f(ln(Pr(z_p \mid \Lambda^f)))$	$p \in P^l, p \notin F^l \cup B^l$
	H	$p \in F^l$
	0	$p \in F^l$
$\{p, B\}$	$\lambda f(ln(Pr(z_p \mid \Lambda^b)))$	$p \in P^l, p \notin F^l \cup B^l$
	0	$p \in F^l$
	H	$p \in F^l$

Table 2.1: Proposed weight assignment for the graph

task. The Expectation-Maximization (EM) algorithm [10] is used to determine the maximum likelihood parameters which define a mixture of k Gaussians in the feature space:

$$\Lambda^i = \{\omega_k^i, \mu_k^i, \Sigma_k^i\}_{k=1}^{\mathcal{K}} \tag{2.11}$$

where:
k: this is the index for each Gaussian composing the mixture.
$\omega_k, \mu_k, \Sigma_k$: these parameters denotes the weight, mean and covariance matrix obtained for each Gaussian.
$\mathcal{K}$:this is the number of Gaussians which models the input data, in our case, the foreground and background color distributions. We have estimated that the most suitable value for our purposes is $\mathcal{K} = 5$ This value $\mathcal{K}$ is of great importance in accurate representation of a given input. Ideally, $\mathcal{K}$ represents the value that best suits the natural number of groups present in the input. The suitable value for $\mathcal{K}$ is based on a trade-off between performance and number of parameters used for describing the mixture distribution.
The EM algorithm will be the iterative method to obtain the parameters set of the GMMs given its current estimation in two steps. Using EM, the parameters representing the Gaussian mixture are found. A brief description of the EM algorithm is explained in [12].

Global fitting is a time-consuming process. The time required to assess the GMM on the entire video sequence is a limiting factor for the length of the sequence that can be modelled with this technique in one step. In order to find out the way to avoid this limitation we have studied other algorithms to perform the same task.
The Improved Fast Gauss Transform (IFGT) aims to efficiently estimate sums of Gaussians in higher dimensions, where a new multivariate expansion scheme and an adaptive space subdivision technique dramatically improve the performance [1]. The improved FGT has been applied to the mean shift algorithm achieving linear computational complexity. It would be possible to develop a future version of our system according to this new way to face this part.

Finally to sum up, according to the defined energy function, the weights of the graph edges which translates the meaning of the segmentation problem into the graph cut are summarized in Table 2.1.

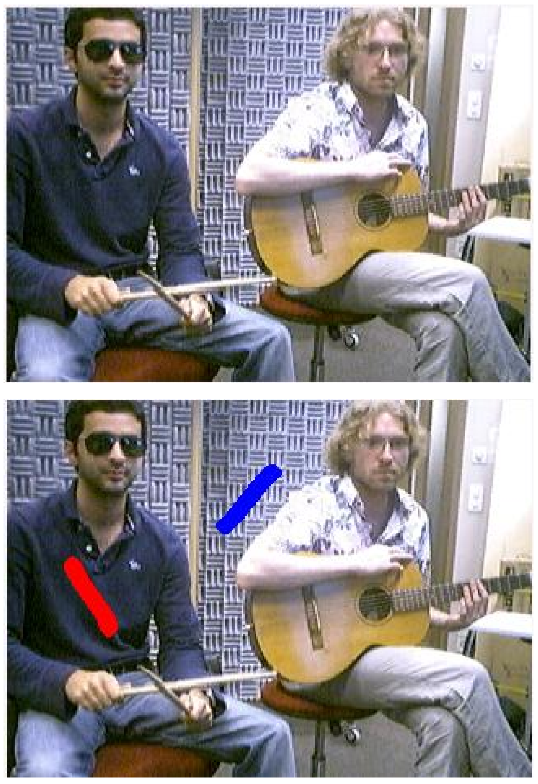


Figure 2.1: One frame extracted from a sequence of the kind of video signals we work with and its corresponding labelling provided by the user; red scribbles are sources (seeds which identify the foreground) and blue scribbles are sinks (seeds which identify the background).

2.4 Methodology: solving the problem

Once we have stablish our video segmentation problem and its features we will show the methodology that we will follow to solve it.

Our algorithm requires a video-signal as input.
As we explained from the beginning, our method is a semi-supervised process.This means a user is the responsible for setting the initial data.
Either seeds or scribbles have to be provided as hard constraints to identify foreground and background labels. At least there must be one of those pixels in each region as the main point and therefore we will link the other ones to them with a harder weight as we state before because we a priori know they belong to its region with no doubts.
These clusters of pixels can be set in any frame of the current group in the algorithm because we work with the video-signal as a 3D structure. We can illustrate this stage of our process with an example in in Figure 2.1.

Given the required information to apply the video segmentation, the video-signal and the labelled regions (foreground/background), it is necessary to focus on how we are going to address the problem. We split the whole video signal into a sequence of blocks of frames (BOF). They are overlapping and identically sized but the last group which can have a different size. This division will provide several benefits to our approach. The overlapping structure of BOF will allow to propagate the segmentation of one group into the next one. Each neighboring pair of blocks share one video-frame. Once the segmentation of one group is achieved, its last frame is shared with the next group to fix its first frame. Then, the segmentation of this frame will be used to set prior information of the labelling as hard constraints. Furthermore we can take advantage of the graph structure. We build the triangular upper adjacency matrix for the graph using sparse representation to save memory and time. The first block matrix is compute and we reused for the whole remaining groups but the last. This let us to save time and operations. Finally we have to highlight this methodology allows to process video-sequences step by step of bigger length, to build a total result. Otherwise it is not feasible to handle a huge amount of pixels because we are restricted by limits of amount of data of our tools (Maximum-flow algorithm only process one million of points as we will show in the practical implementation of our method).

As a starting step of our algorithm we have to compute the structure matrix of the graph as a volume of pixels in the space-time domain. Spatially each pixel is connected to the four neighboring ones, on the top, on the back, on the left and on the right into the same frame. In the time domain each pixel is linked to the one at the same position into the previous frame and the following frame. This has to be stored for next BOFs but the last. It can have a different size so in that case we recompute this.
Next we have to asses the boundary weights for this structure using the boundary term of our approach for RGB intensity of pixels. In addition we set the initial hard constraints for foreground and background, linking them to the main point of the labelled regions. The main point is fixed as the first provided for each one.
Last step of this part of our algorithm is based on computing the regional term of the energy function which give us the video segmentation. It consists of training the GMMs for foreground and background color distributions with initial hard constraints. Therefore the GMMs parameters are estimated. With this information log-likelihood for each pixel is computed following both models, foreground and background. By applying the function which acts over these results to set the right range of its values, we find the appropriate weights. Finally we assign them to the corresponding links to our matrix structure.

Now, we will proceed to perform foreground/background segmentation via max-flow/min-cut energy minimization over our graph.
Therefore the resulting hard segmentation is given as graph cut represented by a 1D binary matrix composed of pixels which fit into the foreground represented by 1s and pixels which fit into the background represented by 0s. Results of applying the cutting mask over the visual content of the input video signal are shown in Figure 2.2.

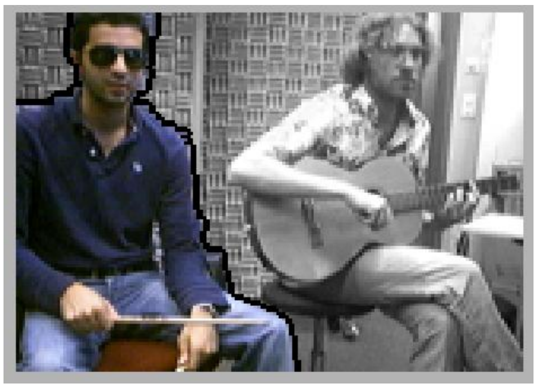


Figure 2.2: Segmentation of one frame: foreground is depicted in the RGB color space; background is shown in gray scale; a black contour is drawn to better observe the difference between them.

As it is explained in [5] our segmentation results either may have isolated regions or may also contain holes. Our algorithm allows very efficient editing of segmentation results, if necessary. The optimal segmentation can be efficiently provided if some hard constraints (seeds or scribbles) are added or removed manually by the user as the initial hard constraints. Our algorithm efficiently adjusts the current segmentation without recomputing the whole graph from scratch. We perform the cut adding only the hard weights of the new constraints to the stored matrix of the last graph. Therefore we obtain a re-segmentation which means an improvement over the last result. This a mechanism to control and supervise the quality of process.

When the desired segmentation is achieved we will proceed to extend the labelling of the regions to the next group of frames. We profit the cutting mask of the last frame of the processed BOF. These will be the first frame with the hard constraints assigned automatically to apply the procedure to next group of frames. To avoid errors, in case labelling is not accurate enough, we will build a trimap. We will dilate and erode the segmentation cut and the margin between foreground and background limits is defined as unclassified. Only the labelled values will be assigned. In addition, training regions for foreground and background color models will be assessed according to the segmentation results. Video tubes composed of all the previous BOF already segmented will be used to set the training points to better estimate color GMMs with more information. These video tubes will be dilate and erode as we did to define the initial data for each BOF. Thus we will allow again an unclassified region in order to avoid errors at the segmentation borders. Thereby we are able to estimate more accurate GMMs for the color distributions of our regions. Finally we are able to perform the propagation of the segmentation.

Following this methodology over the total amount of BOF which compose our video-sequence we can achieve the total video segmentation. We have to point out this can be performed, applying the propagation procedure over each BOF step-by-step or over the total succession of BOF in one step. Propagation step by step allows the user to supervise the process, improving the segmentation after each step. However segmentation one step is an automatic and uncontrolled process.

Chapter 3

Link between audio and video channels

3.1 Introduction

The objective of this approach is to extract an audio-visual source from a video-signal. To reach our aim we have started from the separation of our targets from the video signal. We have achieved to perform a segmentation method to split it into two regions labelled by the user. Therefore the original signal is now classified as a foreground (the selected source) and a background (the remaining content).

Considering this condition we have to take advantage of the video information to find the method to translate the segmentation into the audio track.

Intuitively for the kind of sequences we work with, an audio event is associated to a visual movement of its corresponding source. To illustrate this we can think about a guitarist who moves the hands to produce the sound or a person speaking or singing who moves the lips to originate the audio signal. This meaningful fact can be profited to stablish the link between audio and video channels.
In our approach this will be carried out by extracting the motion of the video signal correlated to the audio track.
These results give us the key to find out the periods of audio activity of the signal for the sources we have. Finally processing this data, we will be able to estimate the separation of the audio signals.

3.2 Audio-Visual Diffusion

3.2.1 Introduction

As we can read in [?] Audio-visual analysis aims to put together information coming from audio and video channels. The aim of this section is to perform a method to extract video movements related to the sound. Therefore, using the segmentation information of the visual content we will be able to translate this into the audio signal and find out the audio activity which belongs to each labelled part.

The procedure we have performed over the video signal is homogeneously diffusion of parts whose motion is not coherent with a synchronously recorded audio track. Therefore, video movement related with audio events will be enhance and located into the diffused signal whereas not synchronous motion should appear eroded in this resulting signal. Thus when in next section we work with the video amount of motion corresponding to the audio events ,only the right information will be taken into account.

In our algorithm we will take advantage of the method developed in [?] which is able to develop a non-linear video diffusion process controlled by a diffusion coefficient that it is a function of the synchrony between audio energy and video motion at each point of the video signal.

3.2.2 Describing the procedure

We start the process with our RGB video sequence $\upsilon(x, y, t)$ defined as a function w such as:

$$w : \Omega \longrightarrow \Theta$$

$$(x, y, t) \longrightarrow w(x, y, t) = \upsilon(x, y, t)$$

where:
Ω := X x Y x T: this is the space-time domain where our video sequence is applied.
Θ := R x G x B: this is the 3D domain for a video signal in the RGB color space.

In order to formulate the video diffusion problem we need to manage a function defined by:

$$f : \Omega \longrightarrow \Upsilon$$

where:
Ω := X x Y x T: this is the space-time domain where it is applied a monochrome video sequence.
Υ: this is the 1D domain for a monochrome video signal.
Thus, we will convert the initial RGB color information into its luminance to work with all the information of the video signal applied to 1D domain.

Then, following the formulation explained in [**?**] the problem of video diffusion for a monochrome video signal is defined by the equation:

$$\delta_\tau \upsilon = div(D\nabla \upsilon) \in \Omega x[0,\infty] \tag{3.1}$$

where:
D:this is a positive definite diffusion coefficient.
τ: it refers to the diffusion time.
t: this will be the temporal axis of the video signal.
div(), ∇: they denote respectively the divergence and the gradient operators with respect to the space variables. The problem will be completed setting the boundary conditions to:

$$\upsilon(r,0) = f(r) \in \Omega \tag{3.2}$$

$$< D\nabla\upsilon, n >= 0 \in \Gamma x[0,\infty] \tag{3.3}$$

where:
$r = (x, y, t)$: these are the 3-D video coordinates.
n: it is the outer normal.
$< .,. >$: it is the Euclidean scalar product on $\Re^3$.
$\Gamma = \delta\Omega$: this is the boundary region of the video signal domain. This equation (4.1) belongs to a general class of equations which satisfies for a $\tau \in [\tau_0, \tau_1]$ all the maxima of a solution are to be found in the boundary Γ or at $\tau = \tau_0$. This is due to the boundary conditions and provided the diffusion coefficient is positive.

Therefore we must chose the most suitable value of the diffusion coefficient for our purposes. We define this parameter according to the approach in [**?**] as a scalar valued function which depends on the involving video signal itself and this can be written as:

$$D(r,\tau) = g(s_\sigma(r,\tau)) \in [0,1] \tag{3.4}$$

where:

s_σ: this is a regularized measure of the synchrony between audio and video channels; it is defined as:

$$s_\sigma(r,\tau) = (a(r) \cdot \delta_t \upsilon(r,\tau)) * G_\sigma(r) \tag{3.5}$$

where:
$a(r) = a(x, y, t) = a(t)\forall x, y$: this is the energy of the audio channel at time t.
$\delta_t\upsilon(r,\tau)$: this represents the temporal derivative of the video signal.
G_σ:this is a 3D Gaussian of variance σ^2 and the convolution with this function makes the audio-visual synchrony measure much more robust. In our work we will fix σ^2=0.3 because this value has been estimated the most suitable for our purposes.
Therefore the audio-visual synchrony s_σ assesses the coherence between both modalities according to the relationship between audio energy and video motion

at each point of the video volume. High values of this parameter indicates matching for important acoustic events with relevant pixels motion. Otherwise we will have low values close to 0.

$g()$:this is the intensity of the diffusion process which acts over s_σ. Taking the appropriate expression discussed in [?] $g()$ is defined by non-negative monotonically decreasing function which can be written as:

$$g(s_\sigma) = \frac{1}{1 + (\frac{s_\sigma}{\beta})^2} \tag{3.6}$$

Here β this is the parameter which sets the threshold for the diffusion and so it should be chosen carefully; observing its histogram most of the values are close to 0 because the main part of the video motion is not synchronous with the audio signal. For our purposes we have estimated $\beta = 99.5\%$ of $s_\sigma(r, 0)$. This means 5 pixels over 1000 are considered as interesting for this approach. Therefore if there are points with synchrony between audio and video modalities $s_\sigma < \beta$ they are strongly diffused while the rest is less diffused.

According to this parameter we can study the behaviour of the intensity diffusion process applying g as it is done in [?]:
Regions with s_σ low, are regions of video inactivity (static video regions), audio inactivity (silent time slots) or audio-video incoherence (audio and video signal not synchronous). As a result, applying g we obtain a maximal and constant value to 1. This means the diffusion coefficient D is equal to 1. Thus, the diffusion equation becomes to the heat equation. The result is the homogeneously diffusion of the video sequence.
On the contrary, regions with low values of g due to a high audio-visual synchrony s_σ, undergo a negligible diffusion process, it is stopped.

Once we have established the formulation of our audio-visual diffusion problem we have to face the mechanism to compute the solution. In order to solve it numerically with our software we have to discretizate our equation system. As it is shown in [?] to achieve our objective we follow a forward finite difference scheme.
At each point and at each iteration the intensity of the video signal depends only on its previous intensity and the intensities of the six closest spatial neighbours at iteration n. The contribution of each spatial neighbour is determined by the interpolated diffusion coefficient. Concerning the rest of the studied boundary value problem, the original video signal is used as initial condition in equation (4.2), and the boundary condition in equation (4.3) has been respected by setting the diffusion coefficient D to zero at the video boundaries.
To ensures the positiveness of all coefficients which involves the discretization process we have to choose a grid spacing for the diffusion time discretization $\Delta_\tau = 0.15 \in [0, 1/6]$, and the grid spacing for the space discretization is fixed to h=0.7.
Under those conditions, our discretization satisfies the maximum principle introduced before and becomes the tool to solve the problem.
For further details please refer to [?].

Finally we will have to fix a stopping criterion for our diffusion method to stop our process. Attending to the discretization performed over our diffusion equation we can define:

$$M^n := \Sigma_{p \in P} \mid \delta_t v_p^n \mid \tag{3.7}$$

which is the amount of motion M in the video signal at iteration n in accordance to the discretization step Δ_τ: $\tau = n\Delta_\tau$. Here we have:
$\delta_t v_p^n$: this is the temporal derivative of the video signal at pixel p and iteration n.
As we can observe in our procedure the amount of motion in the video-signal decreases through iterations and it converges towards a more or less stable value in which the motion non-related with the sounds in the audio track is mostly eliminated. Then the motion reduction is given by:

$$\Delta M^n := \frac{M^{n-1} - M^n}{M^0} \tag{3.8}$$

which is the amount of video motion eliminated in each iteration of our algorithm.
As we have estimated a reduction of 0.5% does not change the motion map in a significant way so it is useless to waste time to compute another iteration. Due to this reasoning we set:
$\Delta M^{n_{stop}} < \epsilon = 0.005$
as the criterion to stop our diffusion process. Furthermore we have fix the maximum number of iterations allowed to 60 to try to reach our aim if the stopping criterion in not achieved yet.

3.2.3 Applying the method

At this point the global algorithm of audio-visual diffusion is explained and it only remains to point out how our method proceeds to get the result.
We have to notice that all the equations applied to find the solution must be numerically discretizated according to the methodology we have indicated before.

As we stated at the beginning we start with a video-sequence in the RGB color space which we convert into its luminance in order to obtain the proper monochrome video signal for our process and we compute its temporal derivative, also required for our process. Furthermore we have to extract the audio energy from the audio channel to apply this information to our method.
With this information we are ready to solve the steps towards the solution.

As a first step we measure the synchrony between audio and video modalities $s_\sigma(r, \tau)$ obtained by the corresponding convolution with the Gaussian function defined by its variance. Then over this sequence of values we apply the intensity diffusion process $g(s_\sigma(r, \tau))$ to obtain the diffusion coefficient $D(r, \tau)$. Thereby we are ready to solve the diffusion equation and compute the audio-visual diffusion according to the stopping criterion or the maximum number of iterations.

Therefore applying the proposed procedure to an audio-visual sequence we carry out the iterative diffusion of the video movement which is not related to the audio energy. This will let us to analyze the motion of this video signal to set the link between audio end video channels, taking only into account the proper motion events. As we read in [?]

> "Indeed, spatio-temporal edges situated in regions whose motion is not coherent with the audio channel activity are progressively smoothed since those regions are affected by linear 3-D diffusion. As a result, the variation of the pixels intensity across frames decreases and the video motion is reduced. In fact, by observing the resulting video motion after some iterations of the proposed method we can discover where our algorithm places its attention, that is the possible location of the sound sources in the image."

Finally, to sum up, the result of our process can be illustrated and therefore better understood observing Figure 3.1. Fist image is a monochrome frame which belongs to a video sequence where a boy is playing the drumsticks and the other person is playing the guitar. As we can guess there are two sound sources with their corresponding movement. Therefore, when we apply the video diffusion, we can observe we get a diffused video signal. It only shows non-diffused intensity content for the location of the video movement which is synchronous with the audio channel. It is highlighted in the second image with circles.

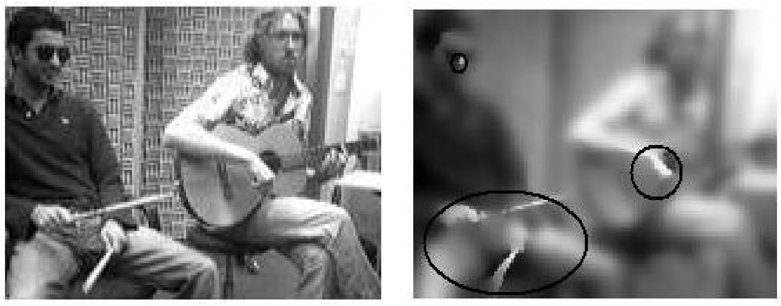


Figure 3.1: Example of the video diffusion process applied over a monochrome video signal; first figure is a monochrome frame of the original signal; second figure corresponds to a frame of the diffused video which is computed from the original one.

The conclusion we can extract from this information is the method proceeds as we expect in regions whose motion is synchronous with audio events. This can be checked by the movement of the drumsticks and the hand which plays the guitar. On the contrary we can observe a non-diffused movement over the nose of the boy playing the drumsticks. This is the example of the main problem estimated about the performances of this method. Video regions whose motion is correlated with the audio track although they are not associated with sound sources, they are not diffused. This the distracting motion we do not want to find. This is related to the threshold β which we fix to estimate the synchrony between audio and video modalities. Although this parameter have been set to the proper value for our approach, this situation can occur. Therefore, when we apply this useful procedure we have to find out the way to overcome this limitation according to our purposes

3.3 Detection of periods where sources are acoustically active/inactive

3.3.1 Introduction

To face the step of audio separation of the sources included in the method of audio-visual extraction of a target that we want to provide, we have to take advantage of the link between audio and video channels developed in our last task.
The audio-based video diffusion we have performed provides the key to do that. By means of this method we obtain the estimation of the motion of the video signal which fits into the audio track. Therefore we have created a diffused video signal where audio behaviour can be extracted.
From the diffused video, it is possible to obtain an accurate audio-visual motion map. It allows to compute the amount of motion into each frame of our video signal which is translated into audio activity of our audio track. This is the data we want to extract as the information which allows to identify the audio sources, learn their models and separate the audio track.

3.3.2 Audio-visual motion map estimation

Once we have develop a segmentation of a video signal into two regions labelled as foreground and background, we need to find out the method to apply this separation to our audio track. With this aim we have performed an audio-based video diffusion which provide us a video signal where un-synchronous motion with the audio is "smoothed" and the rest is "preserved". Therefore the audio behaviour for each label can be clearly identified into the video signal.

Next step of our algorithm is to apply a temporal gradient over the diffused video in order to provide an audio-visual motion map. As a quality measure the temporal gradient is also computed over the original video signal. Thereby we will be able to assess the improvement added by the video diffusion to put together audio and video information of the signal . Therefore it shows how it improves the detection of the acoustic activity of the audio tracks. The obtained audio-visual motion map kept the visual content of regions characterized by audio-visual movement along the time, whereas visual information of static regions is removed. This can be observed in Figure 3.2, as well as the resulting temporal gradient of the original video signal for a frame of the same sequence we have analyzed its diffusion. From these images we can extract the audio-visual motion map achieved over the diffused video, shows more accurately the audio-visual motion and its localization into the video signal than the result obtained for the gradient of the original file. This is because the diffusion process provides a signal with higher contrast between motion “events” than the simple one.

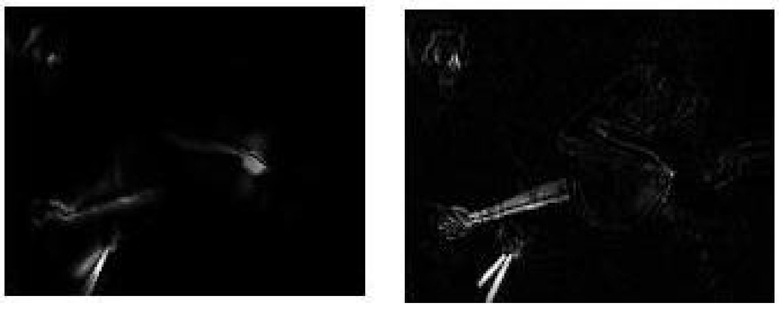


Figure 3.2: Comparison of the audio-visual map obtained from the temporal gradient of a diffused video signal (left figure) and the temporal gradient of the original video signal (right figure).

With the audio-visual motion estimation of our sequence described by its map, now we compute the total quantity for each video frame adding the intensity values of its pixels. As a result we obtain the amount of audio-visual motion which will be the data to extract the audio information.
With this aim, the obtained cutting mask for the video file (previously dilated to avoid boundary errors) is now applied. Thereby the normalized amount of audio-visual motion which fits into the foreground and the background along the time is extracted.

3.3.3 Detection of periods of audio activity/inactivity

At this point, taking advantage of the useful information provided by the audio-visual motion map, we can develop a process to find out the periods of activity for each one of the audio-visual sources of our signal. Before applying next method is necessary to remind the two requirements which our audio-visual sequences have to present in order to be suitable for the audio separation. They are essential to understand the feasibility of next procedure. First condition our signal should respect is at least one audio-visual component belongs to the background region of the audio-visual signal. It allows us to establish the activity/inactivity of the sources by comparing the audio-visual motion of sources. Second feature which our sequence has to provide is audio-visual sources are not mixed all the time. This condition will let us to detect periods during which they are active alone.

The method we aim to find is the procedure to estimate periods where audio-visual sources are active/inactive according to the audio-visual motion. There-

fore, if we analyze the amount of audio-visual motion computed for each of our labelled regions, it provides higher values of audio-visual movement when the audio track is active and lower values when it is inactive. Moreover, the audio-visual motion can present medium-high values when a distracting movement is synchronous with the sound but it does not correspond to any audio event as we explain before.
Therefore, for foreground and background, at the histogram of the distribution of values of the normalized amount of audio-visual motion we can distinguish:
1. Medium-high amount of values over a low point of the motion range. This represents the state of inactivity of the source, because this small amount of movement is very frequent when audio track is off.
2. Medium-high amount of values over a high point of the motion range.It shows the activity mode of the source, because when audio track is on,almost the same high value of movement is repeated.
3. Low amount of values distributed for some points all over the motion range. We have estimated there are two mainly effects which causes these picks of movement. They can be produced by the transitions between the activity/inactivity modes. On the other hand they can be produced because source is moving correlated with the sound but this does not correspond with any audio event. This would be identified as the distracting motion. Both conditions set the process noise which can be defined as "audio-visual motion noise". This formulation can be observed in Figure 3.3. This are the histogram for foreground and background of the sequence which is being analyzed along this chapter. They represent the amount of audio-visual motion obtained for each source. We can appreciated the distribution of values for the three stages we explained before.

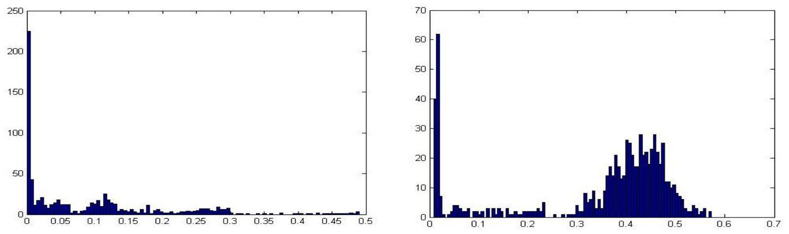


Figure 3.3: Example of histograms for foreground (left figure) and background (right figure) audio-visual motion distributions.

In order to model this behaviour we have assessed that GMM will be again the most suitable tool to evaluate the clusters of audio-visual motion values. To get a better estimation with the GMM we regularize the input signal. We perform a previous smoothing process. We will do an average over a neighboring range for each point of the sequence. In this way, we will obtain a new signal which represents the amount of audio-visual movement for each frame but with less variant values. This will help remarkably to carry out our the current. Thus, applying a 1D GMM composed of 3 Gaussians over the distribution of audio-visual motion values we model the audio states of the targets. Therefore we will have a Gaussian located in low values of the motion range and another situated over high values, representing the inactive and active modes of the audio track respectively. Moreover there will be a Gaussian with lower amplitude which describes the remaining effects we have already explained (transitions between states and distracting motion described as a noise). Figure 3.4 depicts the histograms of audio-visual motion for foreground and background of Figure 3.3 described by means of GMM.

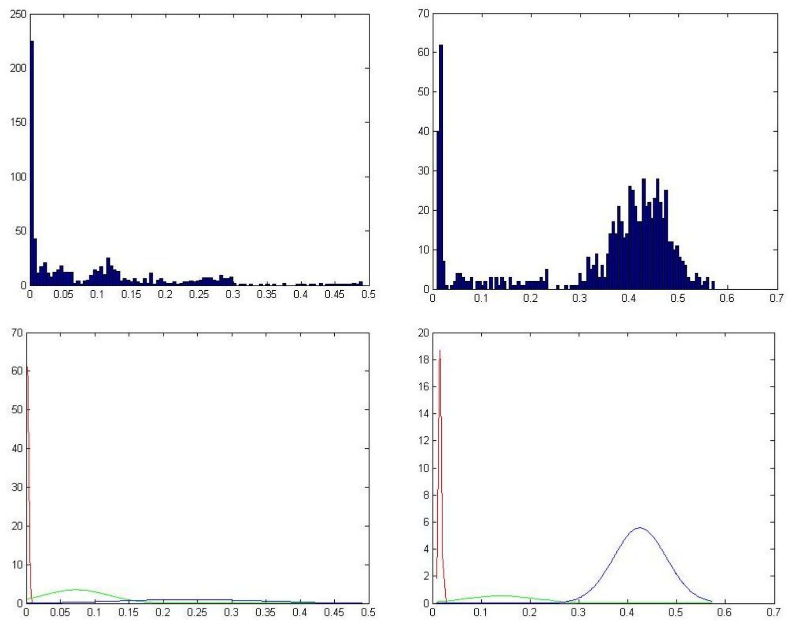


Figure 3.4: Description of histograms for foreground (left figure) and background (right figure) audio-visual motion distributions by means of 1D GMM with 3 Gaussians.

At this point we have to notice GMM is a tool based on estimating the parameters of the functions which compose the model we want. It would be possible the estimation provided fails to fit into the distribution for the data we have. It has been observed in some occasion but we can state the behaviour of our GMM is accurate enough to achieve the successfully results.

Finally working only with the Gaussians assigned to the on/off states we can extract its intersection and therefore obtain the motion threshold to set the activity/inactivity modes for the audio targets discarding the noise. At the intersection the value of these two curves is equal. This means over this threshold it is more probably to belong to the function which models the higher values of the movement (audio activity). In contrast under this parameter we can almost state that values fit into the function which describes the lower movement (audio inactivity). As we can notice with this threshold which identifies the audio activity/inactivity is provided by the audio-visual motion analysis. Thus, the synchronism between audio and video channels have been exploited.

Finally we will compute the time slots (in our discrete case frames) where foreground/background are acoustically active/inactive by means of assessing the the audio-visual motion according to the audio-visual threshold.

Figure 3.5 illustrates the evolution of the amount of audio-visual motion along the frames of the video sequence we are working with for foreground (red curve) and background (green curve). Analyzing this information we can observed the values of the activity/inactivity mode for each source as well as the motion threshold for each one,depicted with a coarse line.

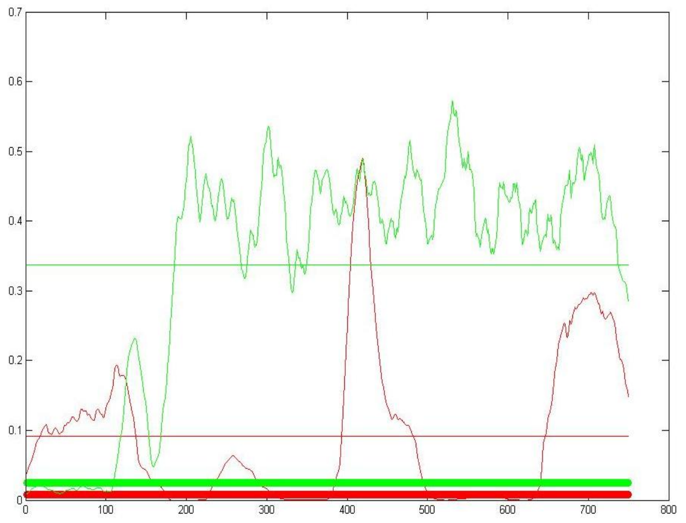


Figure 3.5: Evolution of the amount of audio-visual motion estimation for foreground (red curve) and background (green curve) along the time (frames). Motion threshold for audio activity is depicted as coarse line; the average of values is shown as a fine line.

We have to point out that for our current sequence, we can observe the activity/inactivity modes as the values of the corresponding curve higher than the motion threshold (source on) and lower than the motion threshold (source off). Furthermore, we know the succession of audio events for this sequence. It is composed of a first period where foreground is on, afterwards, it turns off and background turns on, and finally both are on. Figure 3.5 shows this behaviour. But at the same time, we can notice, when foreground should be assigned to be off (because of its audio content) in the middle of the signal, a motion pick appears and turns the foreground on. This is due to a movement correlated with audio track but not identified with audio events. This is a distracting motion which have been discarded to compute the threshold of the states and will be discarded to set the training audio of the signal, aspect which will see in next Chapter.

Chapter 4

Automatic extraction of the acoustic content of the source

4.1 Introduction

At this stage of our method we have established the link between audio and video channels. We have used this information to estimate the periods of audio activity/inactivity for the audio sources identified with the labelled regions (extracted from the video segmentation).

The remaining step to achieve the total objective of our approach is to separate the audio sources according to the obtained information.

This will be performed, extracting the audio samples where each source is active alone, finding the training points to estimate their audio models, and finally we will achieve to separate our audio targets. None supervision of the user is required at this point therefore we can define as automatic this task. Supervision was only essential at the beginning of the problem to label the segmentation regions. Because of this reason our approach provides a semi-supervised method.

4.2 Extraction of training audio samples

In our next stage towards the audio separation we will start with the appropriate information already obtained from our particular processing of the audio-visual signals. This means to profit the information about time slots (in our case frames) where sources active/inactive to set periods where audio sources are playing alone. With them we will be able to extract the audio samples which will provide the training points for next step of audio models estimation

This aim can be achieve when audio-visual sequences respect the condition about the three kind of audio events required for the sequence: at least one period of each source includes its activity alone and one period with its mixture as we explained before. The first step will be to stablish a criterion to assign audio samples to each source with no doubts . In order to be strict, our decision is to assign audio samples to each source for each instant when it is known the target is on and the remaining sound is off. Therefore we will get more accurate results because applying this procedure we are sure that one and only one source is active alone at the assignment moment. According to our initials assumptions for foreground/background labels, this does not means at all, that one audio track is playing alone. In our work foreground is assigned to one audio-visual object of our audio-visual signal but background which represents the remaining audio-visual part, can be composed of one or several audio tracks. Therefore what we aim to extract is the audio track of the foreground from the audio track/s mixture which compose the background.

Thus, we extract the foreground/background audio samples from frames where one source is active and the other is inactive. This process is illustrated in Figure 4.2 following the analysis of the sequence of Chapter 3. Figure 4.1 is the same figure of Figure 3.3 which depicts the evolution of audio-visual motion of the sources along the time, with the respective motion threshold as we explained before. This figure will set the periods of activity/inactivity for each source. According to this information training samples for audio models are extracted for foreground and background when it is estimated they are active alone. We can observe, the distracting motion introduce for foreground, is discarded and audio separation will be achieve successfully due to these accurate training points.

4.3 Estimation of audio models

From the audio-visual processing we have carried out until this moment, we have achieved the segmentation of a video signal and we have already provided almost all the information required to reach our remaining aim for our purposes.

At this stage, with the training audio samples we have extracted in the last step, we will address the audio sources modelling. This is the previous stage to the final separation.
The estimation of audio models for our targets will be performed by means of Gaussian Mixture Model as usually we do in our method to assess robust distributions of different kind of values. Our algorithm follows the method discussed in [14].
Now we work with spectral GMMs because we will apply them to the Power Spectral Densities of the audio channels. This will let us to observe the several frequency behaviours into the same source. This will be an essential information given the diverse nature of the sounds.

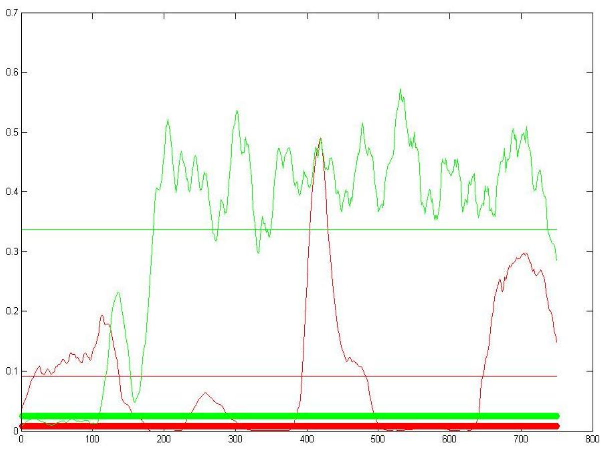


Figure 4.1: Evolution of the amount of audio-visual motion estimation for foreground (red curve) and background (green curve) along the time (frames). Motion threshold for audio activity is depicted as coarse line; the average of values is shown as a fine line.

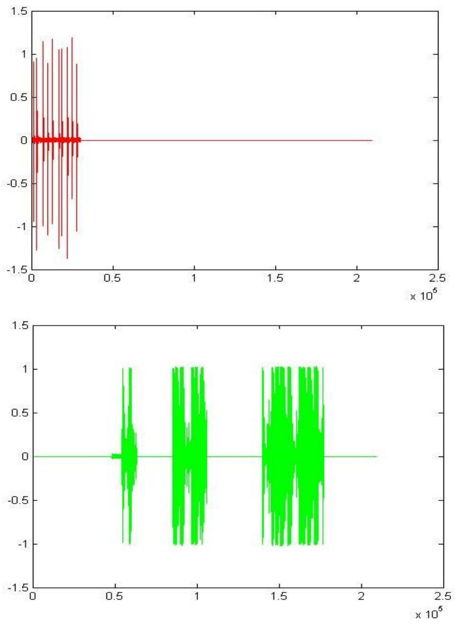


Figure 4.2: Foreground and background audio samples where the respective source is active alone; this data is extracted from the periods of activity of the sources depicted in the figure above; these will be the training seeds to built audio models to complete the total audio separation.

To apply the GMMs, the blind method for audio separation described in [14] requires a training prior information. In order to allow the blindness of the procedure we will extract the necessary data from the available values. The audio samples which describes each audio target active alone, provided from the audio mixture analysis will be this training information.
The learning process for each source starts with this training audio sequence $a_n^{train}(t)$. Next, the STFT (Short Time Fourier Transform) is applied over the training data in order to change to the time-frequency domain, more suitable to handle audio energy by means of spectral analysis. Then a spectral model for the audio source is learned $\Lambda_n^{spec} = \mu_{n,i}, R_{n,i}$ by the maximization of the likelihood $p(\Lambda_{n_\tau}^{train} \mid \Lambda_n^{spec})$. This is iteratively computed by the EM algorithm again.
For further information about the detailed procedure of model learning please refer to [14].

4.4 Audio Separation method

Once we have learnt the audio models from the training data we will perform the audio separation method used and described in [14]. It is adjusted to work with the mixture of two sources as we address our problem but it can be extended to more items. For each time instant it looks for the more suitable couple of states given the mixture spectrum. We will apply the separation method to the audio models, the audio signals and the video probability of the main target as required inputs. It only remains to point out how we set this likelihood.

We will extract the video probability of belonging to the foreground audio track for our problem from the audio periods which describes the audio activity. We set the video probability to 0.75 for the video frames when only this source is on and the remaining sound is off. Thus, at this time slots we are almost sure the active source is our target. Despite this assumption we fix the probability to 0.75 instead of 1 to allow certain margin of error. According to this formulation, on the contrary for the frames when it is known only the remaining audio track is active, we will set the video probability of our target to 0.25. Thereby we express at these moments is very unlikely the sound belongs to our source. Again we allow certain margin of error because we do not fix this value to 0. We will set the remaining audio foreground as unknown. The sound may belong to any source. This is translated to an assignment for the video likelihood of the target to 0.5. Thereby the video probability for the remaining sound is computed as $1-P_{target}$ and thus the logical relations between both probabilities is respected.

Finally we have computed all the required data to perform the audio sources separation, and it can be apply easily to reach our last and challenging objective. As a result we get the separation of both labelled audio sources.

4.5 Quantitative measures

To complete this step which concludes the audio-visual separation of audio-visual sources, we will focus on the quantitative measures we can apply to assess the performances. We will work only with two audio sources, the foreground audio signal and the background audio signal. Although several audio tracks composed our audio signal, one will be assign to the foreground and the rest will be labelled as background. Once we have pointed this out, we can use source to call each one of our both audio targets.
As we can extract from [14] the accuracy of the method applied is highly related to the right estimation of periods for only one source is active. Errors appears when our algorithm identifies one source is playing alone, and the other one is active too. In these error frames the frequency behaviour of the sources mixture will be learnt. Due to this fact, the algorithm leads to a wrong audio separation. Two measures assess the performance of our method in this domain.
The activity-error-rate (ERR) for source S_α represents the percentage of time during which the algorithm makes an important error since it decides source α is active alone and it is not true.
The activity-efficiency-rate (EFF) for source S_α represents the percentage of time in which α is active alone that our method is able to detect.As it is explained in [14] this parameter is very important parameter given the short duration of the analyzed sequences: the higher is EFF, the longer is the period during which we learn the source audio models and, consequently, we can expect to obtain better results on the audio separation part. Table 4.1 describes the definition of these parameters regarding our two sources only. For a general formulation of the parameters please refer to [14]

	Activity-error-rate	Activity-efficiency-rate
foreground	$ERR_{fore} = \frac{F(S_{fore}^{ON} \& S_{back}^{OFF} \mid S_{back}^{ON})}{F_T}$	$EFF_{fore} = \frac{F(S_{fore}^{ON} \& S_{back}^{OFF} \mid S_{fore}^{ON} \& S_{back}^{OFF})}{F_{fore}}$
background	$ERR_{back} = \frac{F(S_{back}^{ON} \& S_{fore}^{OFF} \mid S_{fore}^{ON})}{F_T}$	$EFF_{back} = \frac{F(S_{back}^{ON} \& S_{fore}^{OFF} \mid S_{back}^{ON} \& S_{fore}^{OFF})}{F_{back}}$

Table 4.1: Quantitative measures for audio detection; ERR: Activity-error-rate; EFF: Activity-efficiency-rate.

S_α^{ON} :=“α is active”
S_α^{OFF} := “α is NOT active”
$F(A \mid B)$ is a function that returns the number of frames where our algorithm estimates that the event A has place and the ground truth soundtracks indicate that the current event is B.
F_α is the number of frames where α is active alone.
F_T is the total amount of frames.

Chapter 5

Graphical User Interface for Audio-Visual Segmentation of a video signal

For our purposes we have designed a Graphical User Interface (GUI). This will be the tool which allows the user to supervise the extraction of audio-visual targets following our approach. We have profited the Graphical User Interface (GUI) software that we can find in Matlab.

Thus, through the GUI, user manually provides the essential information required by our method, the hard constraints which label the regions of the visual content of the sources, setting the foreground and background. The audio part of the targets will be extracted automatically.

The appearance of our interface is depicted in Figure 5.1. We are going to analyze in detail how our tool works through this chapter.

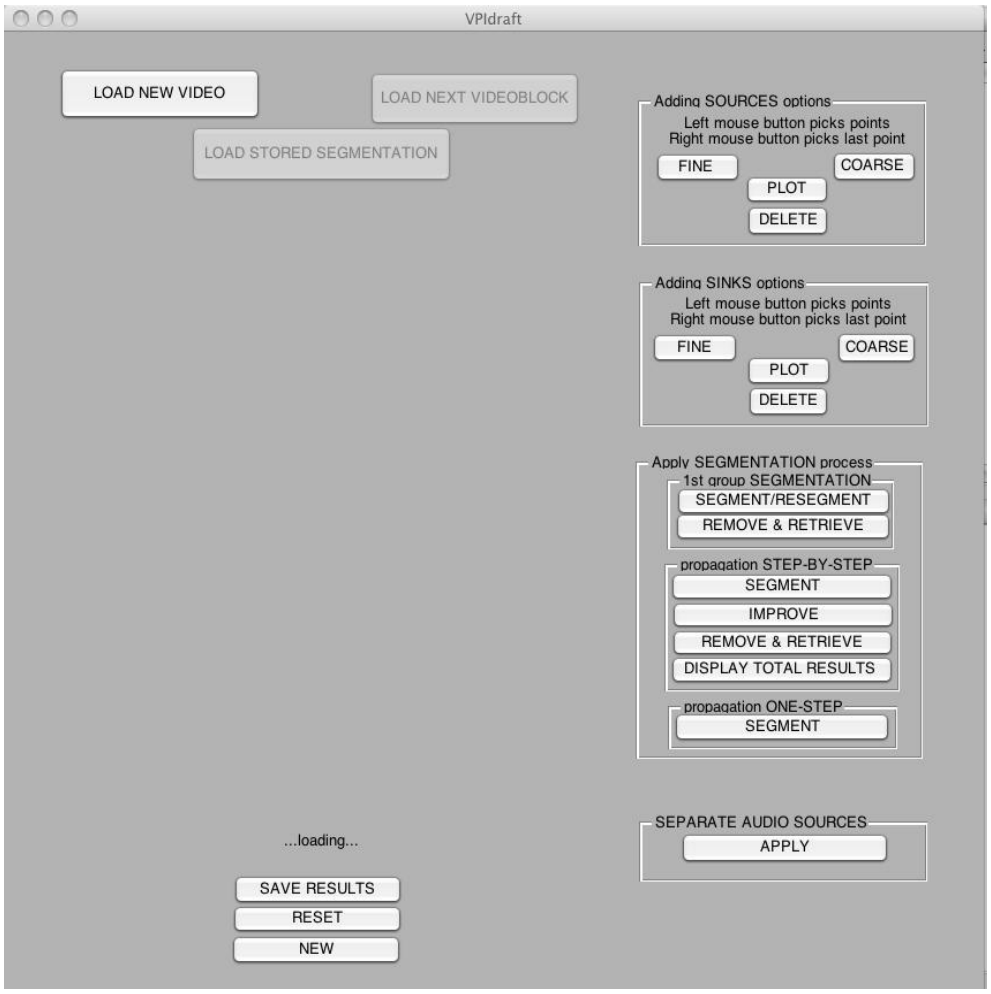


Figure 5.1: Extraction of audio-visual sources Interface

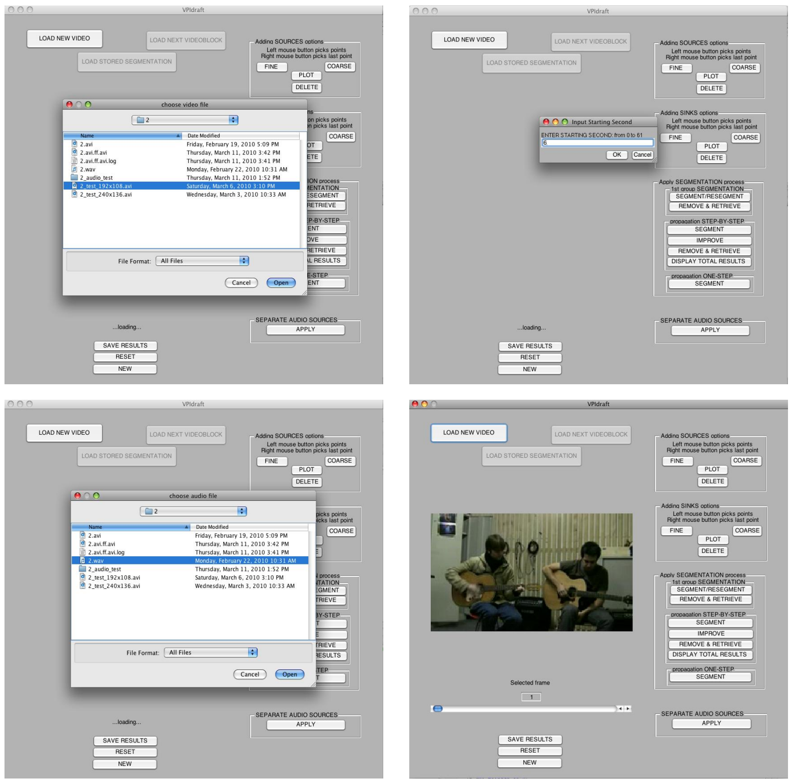
5.1 Loading the GUI

Our method starts loading the input files. The button "LOAD NEW VIDEO" ask the user for providing the audio and video signals. They will be stored in order to work with them when necessary.
The first step of our tool is to extract the audio and video features. This will set the behaviour of our particular audio-visual processing. Mainly, as key points we will find out, the size of our video signal and audio rates and sampling parameters.

Figure 5.2: "LOAD NEW FILE" button: loads the audio-visual files required by the GUI

As we explained in our formulation about video segmentation we work with BOF (Block of Frames). This allows us to respect the limitations of maximum size allowed by the algorithms we apply. We have estimated that we can only assess the Maximum-flow algorithm given by [17] to solve our graph cut problem to a maximum quantity of data about one million of points. This means that the bigger graph we can build is an structure of one million pixels. According to this condition and the image resolution (MxN pixels) of our video input we get the amount of frames that we are able to handle in one step. This fundamental video fragment sets one BOF. With this information, we can get the total length of the input signals we are able to process, given by BOF. It must be fixed from the maximum amount of data that our hardware (Matlab and our computer with its features) are able to work with. We have to point out again, the advantage of working with BOF from a computational point of view. Each BOF has the same amount of pixels, and thus the graph matrix has the same structure, so it is only necessary to compute it once and profit it as explained in our formulation too. Only the last BOF can have a different number of frames, so only in this case will be necessary to recompute it.

Besides these conditions, the length of our audio-visual input signal has to respect other requirements. According to our audio processing formulation and the assumptions we studied about it, it is absolutely necessary to have a changing audio-visual fragment. This means the audio content of our sequence has to be composed of three kind of events: intervals where foreground is active alone, intervals where background is active alone and intervals with a mixture of these events. This is due to the need to provide training periods for each source to reach our audio separation objective. Furthermore the length of these periods when sources are playing alone must be long enough to learn the audio model for each source. To try to respect these requirements the interface will give the option to choose the initial second of the audio-visual input. Therefore this enables the user to select the suitable shifted fragment of the sequence which



Name	Date Modified
2.avi	Friday, February 19, 2010 5:09 PM
2.avi.ff.avi	Thursday, March 11, 2010 3:42 PM
2.avi.ff.avi.log	Thursday, March 11, 2010 3:41 PM
2.wav	Monday, February 22, 2010 10:31 AM
2_audio_test	Thursday, March 11, 2010 1:52 PM
2_test_192x108.avi	Saturday, March 6, 2010 3:10 PM
2_test_240x136.avi	Wednesday, March 3, 2010 10:33 AM

Figure 5.3: "LOAD NEW VIDEO" button features

includes the proper audio events.

All these processing features are automatically assessed by the interface with the unique user interaction of pushing the button "LOAD NEW VIDEO"as we explained from the beginning. It only will ask for the input video-file, the input audio-file and the starting second as we can observe in Figure 5.3. Immediately after this, our software will display the first group of frames into its figure. The user can move along frames, with the slider down the figure. Even it is possible to go directly to the frame we wish to show, writing its number in the text box above the slider which indicates the current frame displayed.

5.2 Video Segmentation using the GUI

Once the interface is loaded and ready to work, by applying the "LOAD NEW VIDEO" button as we explained in the previous section, the video segmentation of the first video fragment already loaded can be carried out.

Previously, our semi-supervised audio-visual extraction process needs the user interaction to provide the seeds/scribbles which set the two regions of our labelling problem. It is a necessary condition to set at least one point as foreground and one point as background before starting as a requirement of our algorithm behaviour. The user should pay attention to this. Other important aspect we have to highlight is the first seed we establish into the foreground or background will be the main seed in each case. Thus the user should be careful and responsible about it. Furthermore, we have to point out the seeds can be set within any frame of the current group. Seeds which label the foreground are called SOURCES and seeds which label the background are called SINKS. The battery of buttons included in “Adding SOURCES”/“Adding SINKS” perform the labelling task with the useful options:“FINE”/“COARSE”,“PLOT” or “DELETE”.

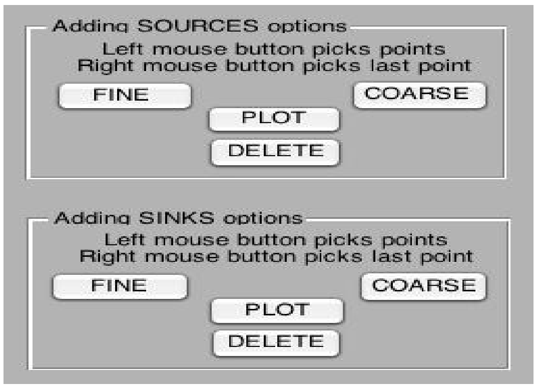

Adding SOURCES options
Left mouse button picks points
Right mouse button picks last point
FINE
COARSE
PLOT
DELETE
Adding SINKS options
Left mouse button picks points
Right mouse button picks last point
FINE
COARSE
PLOT
DELETE

Figure 5.4: Battery of buttons which are the tools to handle the seeds. There is a group of buttons for kind of seed: SOURCES (foreground) and SINKS (background).

“FINE” button allows to draw seeds over the selected frame to add fine contours our regions to the corresponding label. This will be stored as a line interpolated using splines.
“COARSE” button provides the option to depict rough regions as seeds assigned to the chosen label. This will be stored as coarse regions interpolated using splines as done with FINE button. For both cases, the seeds will be shown when the user finish of adding as illustrated in Figure 5.5. “DELETE” button deletes last seeds and shows the current frame without them, none of the labels will appear with seeds.
“PLOT” button shows all the seeds fixed in the current frame. If seeds are off, one push of this button plots the seeds. If seeds are on, one push of this button hides the seeds.

Once the user has performed the labelling of the regions, the interface is ready to apply the video segmentation process over the first group of frames by means of the “1st group SEGMENTATION” buttons.

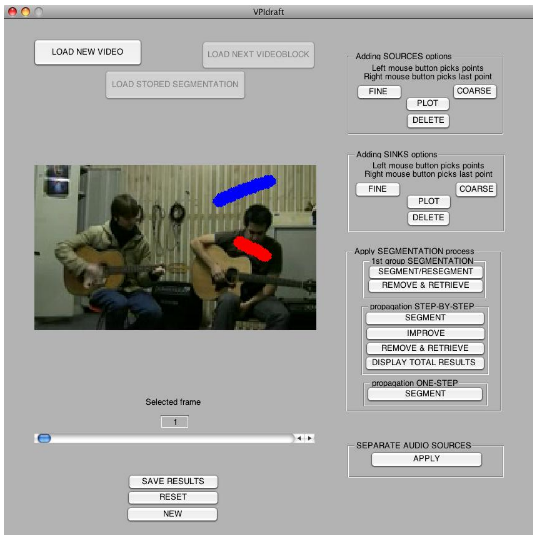


Figure 5.5: Labelling of the first group of frames; sources are depicted in red; sinks are drawn in blue

Figure 5.6: Battery of buttons to process the first group of frames.

The segmentation is performed by the user interaction, pushing the "SEGMENT/RESEGMENT". The underlying algorithm solves the problem following the formulation developed in our work. The interface computes the cutting mask over the sparse graph matrix applying the Maximum-Flow method for Graph Cut of the Boost Graph Library [**?**].
The cutting result is the binary matrix of our hard segmentation. It is composed of 1s to represent the foreground and -1s to describes the background. In order to save memory and to optimize our system it is transformed into logical data. With this aim we will store the background values as 0s instead of -1s. When the segmentation is achieved, it is showed by the interface. The cutting mask is applied over the original video signal and the result is depicted as follows: the segmented foreground is shown in the RGB color space whereas the remaining background is shown in gray scale.
In order to better observe the limits between these two regions, the difference between them is represented with a black contour. These results can be observed in Figure 5.7.

At this point, the fundamental segmentation, i.e., the video segmentation of the first group of frames is now available to work with it. Now, the user can choose

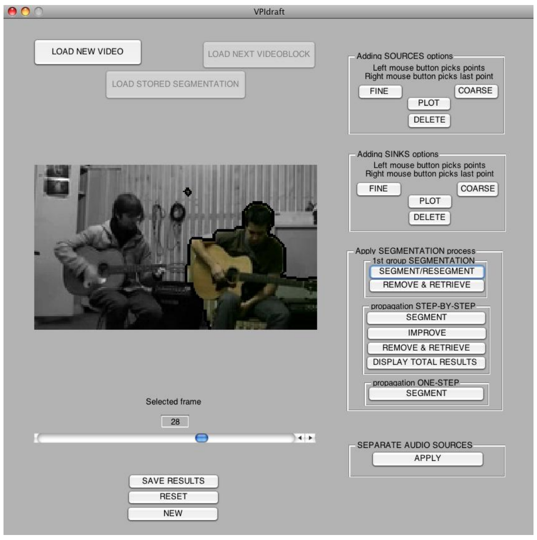


Figure 5.7: Segmentation results of a group of video-frames displayed through the interface.

among three options:
1. to delete the segmentation results and start again.
2. to process the obtained results and improve them.
3. to propagate a appropriate segmentation already achieved along the remaining video signal.
It is possible the user needs to delete this first segmentation, and start again, because the seeds have been depicted erroneously and the user had not noticed it. Pushing "REMOVE & RETRIEVE" button, it will be done.
If the achieved segmentation is acceptable but is not good enough, the user can improve it adding new sources/sinks and use again the "SEGMENT/RESEGMENT" button. Now the interface will find out that we are applying a resegmentation instead of a first segmentation and so the corresponding algorithm will be applied. The main difference between both is that the resegmentation algorithm does not recompute the graph matrix. It only will add the new seeds as hard constraints and the new color information according to this, assessing again the color models for each region and the color information for each pixel. Afterwards if user wants to delete this result and retrieve the previous, it can be done again by pushing "REMOVE & RETRIEVE" button. An example of the previous step to apply the resegmentation procedure is shown in Figure 5.8.

When the desired segmentation for the first fragment of frames is already achieved and the user decides to propagate, two options are available:
1. to perform a manual segmentation step-by-step.
2. to perform an automatic segmentation for the whole remaining video signal.
For both options, the applied algorithm works in the same way according to our segmentation approach. This means, the overlapping division of the total sequence into BOF will be profited. The segmentation of the last frame of the previous group will set automatic labels in the first frame for the next group.

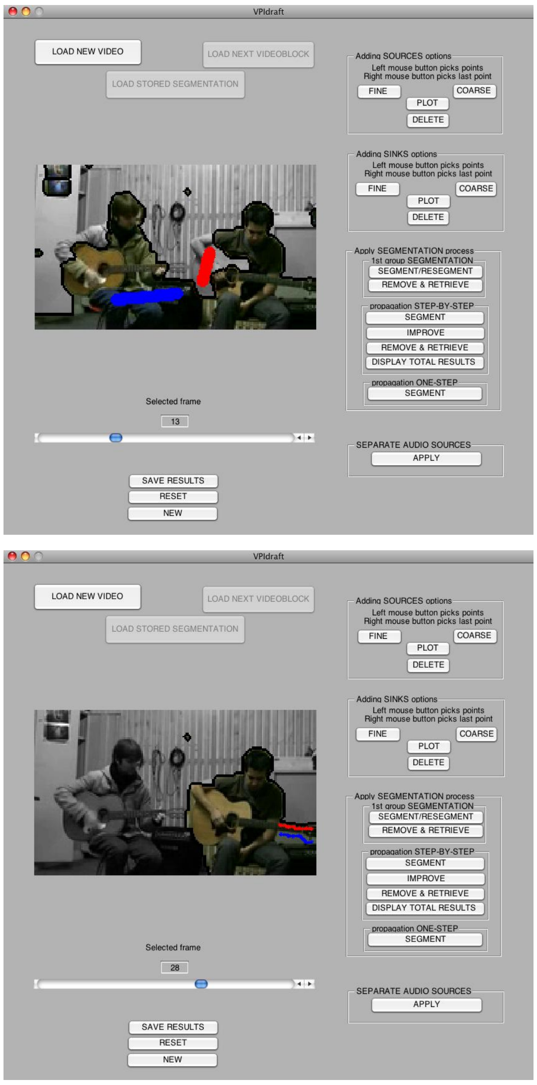


Figure 5.8: These figures describes the previous step to apply the resegmentation procedure by pushing "SEGEMENT/RESEGMENT" button in order to improve the rough result shown; this is the step of adding the corresponding hard constraints, to edditting the results; the figure on the right/left shows the use of FINE/COARSE to add SEEDS

Furthermore, we take advantage of previous blocks segmented to assess more robust color models for our algorithm. In both cases, as we have explained in the corresponding section, to avoid errors we establish a trimap with an unknown boundary between segmented regions whose seeds are discarded to preserves the limits between them.
Thereby if the user decides to do a manual segmentation step-by-step for the remaining video signal the interface features with this aim are the buttons included in “propagation STEP-BY-STEP”.

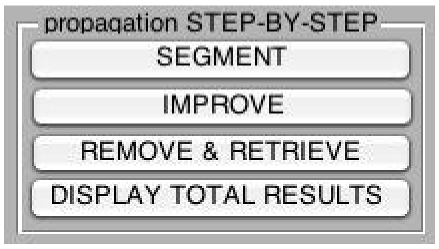


Figure 5.9: Battery of buttons to perform the propagation process by the STEP-BY-STEP method.

To carry out a new segmentation it is necessary to push the “SEGMENT” button.
In this case the option of improving the results will be available too, as it exits for the first group of frames. This will be performed by means of “IMPROVE” apply over the new seeds added as before.
It is possible to delete the current segmentation and to retrieve previous result as before pushing “REMOVE & RETRIEVE” button.
For each group of frames only this video fragment is displayed through the interface. However when the total segmentation is achieved after the step-by-step process, this can be totally shown pushing “DISPLAY TOTAL RESULTS” button.

On the contrary, if the user prefers to apply an automatic segmentation to the totally remaining frames after the first segmentation or at any stage of the step-by-step segmentation, the corresponding button to push is the “SEGMENT”under the category “propagation ONE-STEP”. Now, once the automatic segmentation is computed, it is automatically shown.

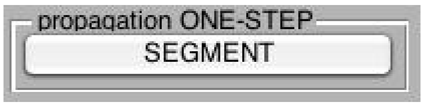


Figure 5.10: Button which performs the propagation of the segmentation along all the remaining video frames in only ONE STEP.

After these steps, when the video segmentation is achieve we will be able to face the audio separation.

5.3 Audio Separation

The user is only required to push "APPLY" button included with the label "SEPARATE AUDIO SOURCES" to carry out all the steps which leads from the video segmentation to the audio separation of the sources. The underlying algorithm automatically applies the several steps towards the audio sources separation.

Figure 5.11: "APPLY" button to perform the separate sources procedure.

First the audio-based diffusion process is performed over the input video signal. This acts over the audio track and the luminance of the input video computed from its RGB components. Once we have achieve the result following the procedure described in the corresponding Chapter, its gradient is computed and the definitive audio-visual motion map is provided as result. This data and the dilated cutting mask will be used to find out the periods of activity/inactivity for the audio sources as we explained in our work. According to this, the algorithm extracts the corresponding audio samples where sources are active alone. This information is used to set the training points to estimate the audio models for the audio targets. Once we have this and following the methodology discussed in our work we get the audio sources separation.

As result, the interface will provide the audio files for each audio source, after the separation.

5.4 Additional features

We can notice there are three buttons under the figure which offers the typical performances of a software: "SAVE RESULT", "RESET" and "NEW" buttons which develop the functions related to their names.
"SAVE RESULTS" saves internal variables of our interface in order to manage this information with certain purposes.
"RESET" performs a total reset of the interface variables this means the video segmentation and audio separation results too.
"NEW" closes the current interface and opens a new instance with all the features reinitializated.

Figure 5.12: Additional button: “SAVE”,“RESET” and “NEW”.

5.5 Future performances

At this point we want to stress that there are two inactive buttons in the upper part of the interface which are features roughly developed and not tested.

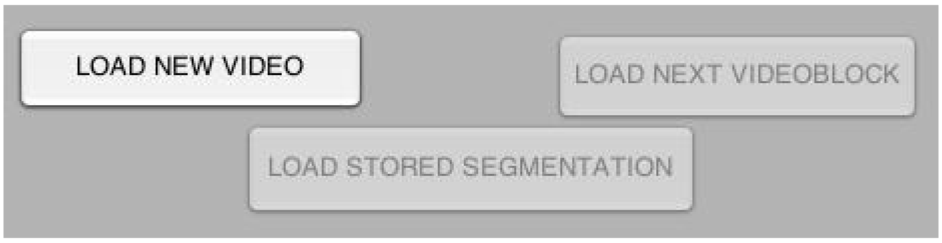


Figure 5.13: Future performances.

“LOAD NEXT BLOCK” button is a possible performance which would let the user to work with audio-visual sequences of any length. Thus, first the interface would work with a fragment of the total input signal as before. Its beginning would be delimited by the starting second of the signal given by the user and its ending second would be fixed according to the maximum amount of pixel hardware allows. Once this amount of group of frames would be segmented we could apply ”LOAD NEXT BLOCK” button to follow the process. This means, if we would push the button, the interface automatically would find out which fragment of the input sequence is the following one for the procedure. Then, this would be set as the current video fragment to work with. When the video segmentation of the desired length is achieved we would be able to apply the audio separation. Therefore we would overcome the limitations of our interface related to audio requirements. To extract training audio samples to estimate audio models for the sources,we need three kind of events in the input audio signal as we explained before. If we are restricted to short sequences, it is more difficult to get suitable audio-signals to work with. This feature would allow us to work with longer sequences, thus it is more probably to find along them, the three kind of events we need to respect the audio input requirements.
“LOAD STORED SEGMENTATION” is a possible improvement of the features of our interface which would allow to load a segmentation previously stopped and saved. It would be possible to load and retrieve it from any point of the

procedure. Therefore it would possible to resume the video segmentation or perform the audio separation.

To complete this chapter we want to introduce another performance which could be developed from our audio processing point of view which has not been included at this moment. It would be a feature which would allow the semi-supervised extraction of the audio part. The interface would have a database with audio models for the typical kind of sources which we work with. Then the user would choose the audio models for our targets as a prior audio information. Therefore we totally outperform the audio conditions required in our method for our signals to achieve our purposes.

Chapter 6

Experiments: Assessing the quality

6.1 Introduction

To assess the performances and the quality of the Semi-supervised extraction of audio-visual sources method of our approach we have performed several experiments. We must explain the steps we have developed to test our method before analyzing the obtained results. The procedure have been carried out by means of the designed Graphical User Interface (GUI).

First step have been to apply the video segmentation process. For each video signal we have started labelling the foreground and background regions by adding seeds over the first group of frames. We have applied the first segmentation and improved it before its propagation. Once we have achieved the suitable segmentation we have extended it along the remaining video signal using the step-by-step method. Therefore we have taken advantage of the option to improve the segmentation of the current block when necessary, adding the corresponding hard constraints.

Once we have obtained the video segmentation, we have applied the separation of the audio sources, performed by its sequential stages. The audio-based video diffusion is applied to our video sequence and after this, the motion map have been computed over it and the temporal gradient of the original signal. In this way, we can estimate the improvement that the video diffusion provides to our audio separation. Then the periods of activity of the sources are estimated over this data. From this information, the audio samples to train the audio models are extracted. Finally the audio models are estimated them and audio separation is achieved according to them.

We have to point out that all the parameters of the total method built in our approach are fixed to the values we have highlighted during the development of

Sequence	*Foreground*	*Background*
3	boy on the right playing the guitar	boy on the left playing the guitar
6a	boy on the left uttering numbers	boy on the right playing the guitar
8a	boy on the left playing the drumsticks	boy on the right playing the guitar
9a	boy on the left singing a song	boy on the right playing the guitar
10	boy on the left playing the guitars	boys on the right singing a song

Table 6.1: Labelling applied to the audio visual sources of the most meaningful sequences of our database which we work with.

Sequence	resolution	length (s)	starting sec (s)	audio requirements	fps	fs(kHz)
3	192x108	28	6	done	25	8
6a	160x120	25	7	done	30	8
8a	160x120	25	12	done	30	8
9a	160x120	25	40	-	30	8
10	192x108	28	3	done	25	8

Table 6.2: Audio-visual features for the analyzed signals.

our work.

The tool of our experiments have been a MAC PC with 2 processors Dual-Core Intel Xeon (4 cores). They have a processor speed of 2.66GHz and system bus speed is 1.33GHz. The Cache memories size per processor is 4MB and the total memory is 4GB.

Our targets are video signals composed of two or three audio-visual sources. The user have decided which region forms the foreground and which region sets the background as we can observed in the Table 6.1. In our sequences the foreground is an object composed of either a person playing an instrument or a person singing or uttering numbers. The background is composed of the remaining elements which do not belong to the foreground. This can include one or more persons doing the same things as into the foreground.

As we can extract from Table 6.2 our database consists of video sequences recorded to test the behaviour of our method. They are resized to a lower resolution proportional to the prior. As a result we work with resolutions of 192x108 or 160x120 . This let us to handle longer video signals which fit into our computational conditions. Thus, according to the resolution and the computational limitation of our hardware, we have been restricted to manage sequences with length of 28s and 25s respectively.
We have to notice that it is very important that our signals include within its length the three kind of audio events needed for the right audio training: a period when foreground is active alone, a period when background is active alone and a period with a mixture between them.
Our video signals respect this specification but one which we was profited to con-

firm the importance of the audio training requirement. With this objective we have shifted the beginning of our audio-visual sequences to the suitable second to respect our audio conditions. This will be trivial for the video segmentation. The video-sequences have been recorded with a rounded rate of 25fps and 30fps respectively, which means we have handled an amount of frames of 700 and 750 frames for each case. According to the maximum amount of pixels which we can apply to the Maximum-Flow algorithm to solve our problem, we divided our video signals. We worked with 43 frames per BOF and 47 frames per BOF respectively. This means we worked with over 1.5s of the signal each time for both kind of sequences.

Once we have explained the framework of our experiments we can start with the several steps carried out to test all the aspects of our approach.

6.2 Video Segmentation Analysis

In this section we focus on the video segmentation of the visual content of the audio-visual sequences. Therefore we have applied the video segmentation for each of the targets defined in Table 6.1, in order to test it and obtain the necessary data to our audio separation.

Performing a time analysis for our method we have estimated the average of computing time to achieve the first block segmentation is about 60s, this means we spend 60s to segment 1.5s of real time video. The improvements performed over this first segmentation experiment a reduction of 20% over the average. This is a small reduction because we not only add the hard constraints to the graph but also we assess the color models taking into account this new data too and this is a time consuming step.
When we extend the segmentation with the step-by-step method, the computing time average for each block is similar to the first segmentation. This is because although the graph matrix is no longer recomputed but the last group of frames, we assess the color models for labelled regions considering information of previous segmentations, in order to extract as much information as possible. However, the improvements over a extended segmentation, are computed within a 30% of the segmentation time, this means a step of that kind takes about 20s. This is because now we only add the hard constraints to the graph structure to fix the new links. Color models are not estimated again because this is time consuming and we have enough information to set the features of our graph.
An automatic segmentation performed after the suitable segmentation of the first BOF, spends about 15min average. This is because the propagation of the segmentation to each block proceeds as the step-by-step method. Then, if the sequence length is 25s/28s and 1.5s are segmented within 1min, the average computation time is 17min. These time features are quite poor despite we have tried to optimize as much as possible the algorithms code and the global procedure. We have estimated, the most time consuming aspects of our method are the Maximum-Flow algorithm and the color model estimation by means of GMMs. To try to overcome these disadvantages we have studied new options to

face the problems they solve. We have searched new algorithms which would be able to replace the former ones. These are a new version of the Maximum-Flow algorithm and different tools to model the color distributions. We will detail these aspects in the next Chapter.

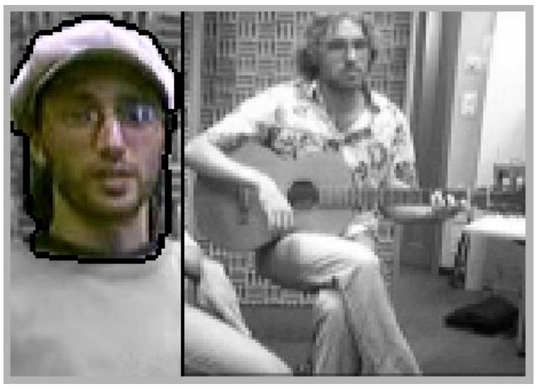
From a qualitative point of view, analyzing our results we can extract, it is easier to cut a foreground from a background when its set of colors is quite different. This means a better segmentation will be achieved with the minimum user interaction. It will be less necessary to add hard constraints to link regions because they will be assigned by its boundary features and its color matching. This has been checked for instance by comparing *Sequence 6a* with *Sequence 3* as shown in Figure 6.1. In *Sequence 6a*, the head of the foreground is composed of colors quite different from the background color distributions. Thus, the proper segmentation have been obtained faster than the others. Only initial seeds and a very few hard constraints were required to obtain the showed result. On the contrary, in *Sequence 3*, we have been forced to add several hard constraints over the first group of frames and improve the propagation of the segmentation to achieve our aim. This is due to the similarity of the components which compound foreground and background color distributions.

Furthermore, we have observed, sometimes segmentation is deteriorated or even “lost”, either along frames or along group of frames. It is possible links between them are not appropriate to keep the connections when pixels move. To face this drawback we have estimated the solution is to add hard constraints in different frames for the same block. Therefore when we re-segment the current group of frames, new seeds are propagated from different parts of the video volume and links are adapted. Due to this, when we apply the automatic method to extend the segmentation, we obtain a rough result because there is no way to control the behaviour of the propagation. This would be an aspect we should study to get a more accurate development of a automatic segmentation.

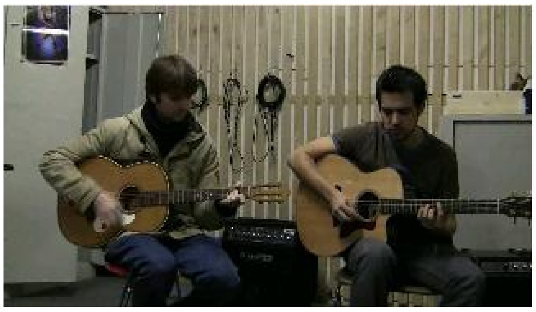
About the interface we have to point out, we have achieved the cutting mask for videos with low resolution. However we can improve the results of our video segmentation resizing the cut and fitting it into a video sequence of bigger resolution. Right now this is something done by the user manually when desired, but it would be possible to study the option to apply this aspect to improve the features of the interface.

These are the main points we can extract about the video segmentation analysis of our approach.

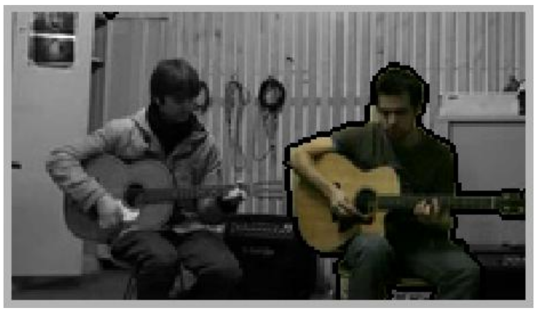
(a) Original frame



(b) Segmented frame



(c) Original frame



(d) Segmented frame

Figure 6.1: Segmentation of a video signal with quite different foreground/background color distributions is described in the figures (a),(b); Segmentation of a video signal with several common components of foreground/background color distributions is shown in the figures (c),(d).

6.3 Audio Separation Analysis

Now we will analyze the results obtained during the several steps followed to achieve the audio separation.

The first step we have carried out towards the audio separation of the sources is to compute the audio-based video diffusion. This represents the link between audio and video channels performed by our method. This profits the synchrony between the two modalities to assess the audio activity over the video motion as we discussed in the corresponding part of our approach.
Therefore we have performed our audio-based video diffusion method over our audio-visual sequences with the decisive parameters of this method set to the suitable values we have estimated and indicated along our study.
However problems appear when our video sequence includes movements in our targets correlated with the audio energy and they will be not diffused. This is the distracting motion described as a “audio-visual motion noise” in previous Sections. This situation represents a challenging condition which we have faced as we explained in its corresponding Section in order to achieve the suitable results.

The obtained results are applied to a temporal gradient as well as the original video signal in order to have a measure to study the improvement provided. By means of the audio-visual motion map obtained from the gradient of the diffused video, we get a monochrome video signal which only represents the intensity of video regions whose motion is synchronous with the audio energy.
This is quite efficient to remove the movement which does not mean anything from our audio-visual point of view.
Anyway this information comes from the video diffused and the distracting motion can appear again.

After this and computing the amount of audio-visual motion for each frame from the audio-visual map we proceed to extract the periods of audio activity for the sources of our signal. This procedure faces the problem of ”audio-visual motion noise” and gets good results for all the audio-visual sequences we have tested. As we discussed in the corresponding Section, the feature of this procedure which solves the distracting motion problem is an extra Gaussian for the components of the GMM model. Two Gaussians models the activity/inactivity distribution values. The third includes the motion values corresponding to the transition between on/off states of the sources and the motion values which appear due to a synchronous motion with the audio energy but not related to the audio activity. This function help us to discard this values for our purposes.
Therefore, we model the activity/inactivity of the sources according to the Gaussians only for the on/off states to establish the threshold between them and achieve good results.
The basic problem found in our detection of audio activity periods linked with audio-visual motion is the relative quantity of movement between targets. If the source of movements for one target is quite subtle, it will very difficult to detect its audio-visual motion according to the video diffused or the original one. This problem will remain and affect our method in the final results. We

have estimated this can be solved with the option we present in our work about to provide a previous information about audio models which will replace the complete training process to extract the audio-models of the signal.

Finally when periods of activity of the targets are extracted we can obtain the corresponding audio samples to estimate the spectral audio models.This will be performed by GMM again and according to these models and their parameters we assess the audio mixture with them.
Finally we achieve the separation of the audio foreground and background.
At this point we have to highlight we have performed an spectral audio modelling of the audio signals. This implies an drawback when we carry out the separation of audio sources with overlapping spectral components. It is a challenge to work with this kind of signals and the results are quite rough. Anyway we have tried to experiment this situation to assess the limits of our methods.

Finally we will analyze our more meaningful results with the suitable figures. *Sequence 3* presents the challenge of including video motion correlated with the audio track into the foreground. Two boys are playing the guitar alternatively and together. In a certain moment for foreground, the boy on the right moves the arm to get something and put it under the guitar. Unfortunately this is a synchronous movement with the sound and it is not diffused as we can observe in the audio-visual motion map. However our method to extract the audio activity and set the activity threshold is powerful enough to neglect this movement which is modelled as noise and therefore we obtain the suitable results.
As we can see in the figures, the Gaussian functions which model the audio-visual motion distribution values for on/off states are allocated in its right positions and they are not affected by the disturbing amount of audio-visual motion detected and included in the noise model. Therefore the evolution of the amount of audio-visual motion along the time shows the right activity threshold superimposed. According to this the right samples for audio training for foreground are extract.

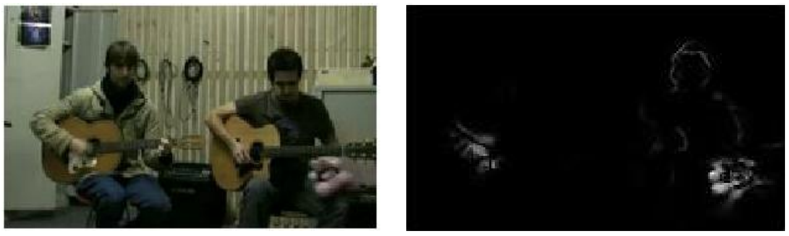


Figure 6.2: Sequence 3: Video frame of the original signal vs. Video frame of the audio-visual motion map.

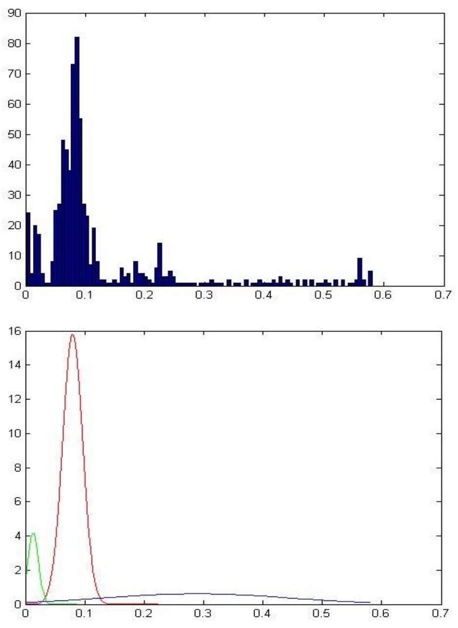


Figure 6.3: *Sequence 3*: Histogram of the amount of audio-visual motion values for foreground vs. GMM estimation for the distributions of the amount of audio-visual motion values for foreground

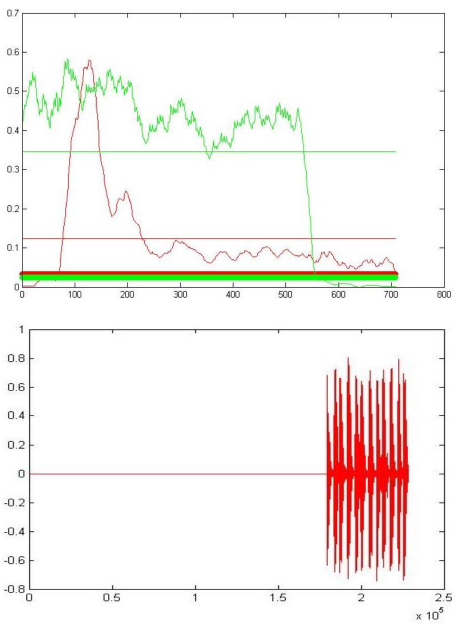


Figure 6.4: Sequence 3: Evolution of the audio-visual motion along the time vs. audio training points extracted for the foreground (red curves)

Sequence 6a is composed of a boy uttering numbers and a boy playing the guitar. For this signal one problem have been to detect an insignificant motion corresponding to the mouth movement into the foreground. Furthermore, the spectrum of the guitar presents problems of harmonics with the voice sound and thus poor results are obtained for the separation of these audio tracks. Furthermore the distracting motion appears because of the movements of the head of the foreground but this will be overcome by our formulation as we have proved with the analysis of the previous sequence. These aspects can be verified observing the negligible motion estimated into the audio-visual map for the mouth of the boy on on the left. This low amount of movement for foreground can be appreciated into the figure of its evolution along the time. This is translated into few periods detected to extract the training points, therefore is more difficult to learn the model with the short length of this signal. The problem of the harmonics should be verified listen the obtained audio files, but the results are a guitar audio track which includes fragments of the voice signal.

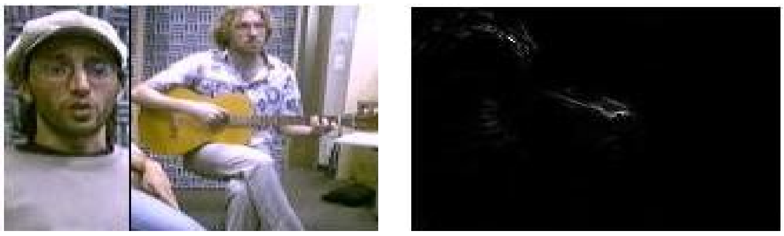


Figure 6.5: Sequence 6a: Video frame of the original signal vs. Video frame of the audio-visual motion map.

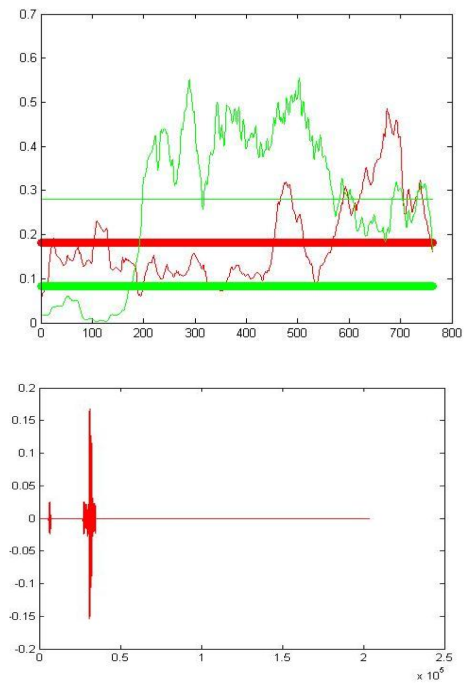


Figure 6.6: Sequence 6a: Evolution of the audio-visual motion along the time vs. audio training points extracted for the foreground (red curves)

Sequence 9a is similar to the previous one. Now the targets are a boy singing a song and a boy playing the guitar. Besides the limitations analyzed for its analogous signal, in this case, the length of the audio-visual signal does not include a period where guitar is playing alone. Due to this training audio samples are extracted erroneously and the audio model for this target is learnt from the mixture. The solution for this would be to allow the longer duration of sequences or provide a database with models for audio signals.

Finally for *Sequences 8a and 10*, audio-visual sources are perfectly extracted, because they satisfies all the requirements and confirm the feasibility of our method.
Sequence 8a is the audio-visual signal provided to the methods of our approach to explain the process and show its performances.
Sequence 10 is similar to *sequence 3* but a background compose of two audio sources and without the distracting motion problem.

Furthermore, we have assessed the quantitative measures defined for the audio activity detection, which are shown in Table 6.1. They express the feasibility of our method with low values and even 0 for the Activity error rate. This means a very low percentage of frames erroneously detected when sources are playing alone. The percentage of right computed frames from the total of them, is high for the main part of our sources and their rates are enough to achieve the suitable results.

	$ERR_{fore}(\%)$	$ERR_{back}(\%)$	$EFF_{fore}(\%)$	$EFF_{back}(\%)$
Sequence 3	1.98	0	100	39.66
Sequence 6a	0	4.1	21.33	76.35
Sequence 8a	0	0	76.22	50.97
Sequence 10	0	13.86	66.22	100

Table 6.3: Quantitative measures for audio sequences; *Sequence 9a* does not appear because does not respect the requirements for the suitable audio detection

The average processing time is not optimized yet, and present an important fluctuation because of the iterative algorithm used to provide the prior information for this process.

Chapter 7

Conclusions and future research directions

In this report we have faced the challenge to develop a Semi-supervised method to jointly extract audio-visual sources from a sequence.
As we have described during our work, the main targets of our method are videos composed of musicians, singers or other sources of sound which would be interesting to process and extract in order to provide them separately or apply them to other signals to compose new audio-visual sequences.

The algorithm starts with the user interaction to set the prior information for the video segmentation. The visual content of our audio-visual sequence must be labelled into two parts: the desired object which defines our foreground and the discarded content identified as background.

In order to achieve the suitable results of our audio separation goal, two requirements are imposed to the audio visual sources: at least one is in the background and at least they are active alone and mixed once.

The audio separation will be provided by an automatic process, thus, it is not necessary to set prior data for our purposes.

The video segmentation of the selected regions can be formulated as a question of finding a cut over the video signal modelled as a graph [?]. The features of the problem will be identified as the minimization of an energy function [7] which describes the boundary and regional information characterizing the pixels connections. The boundary term describes the neighboring relations and the regional term defined by a novel approach adds the color information which is modelled by a GMM estimation. Then a Graph Cut technique will be applied, solving the problem with the algorithm known as Maximum-Flow/Minimum Cut [7]. The procedure applies a contribution which consists of splitting the sequence into groups of frames sharing the previous and last of them. This aspect allows to perform a first segmentation and its following propagation taking the prior data from the shared information. Furthermore, it is possible to improve any result applying a re-segmentation with new labelling data added by user interaction.

This task is carried out by means of the performances of the interface. Thereby successfully results are achieved, but some consideration has to be done. The segmentation and its propagation along groups of frames are time consuming steps. Furthermore when they are poor at first step, the user interaction is required to correct them by labelling new points with the region where they belong. Then the process becomes a hard task because it spends much time. It would be interesting to improve this aspect to optimize this part. One option we have considered is related to the algorithm performed over the graph. As we explained in Section 2.3 it is possible to update the segmentation cut by adapting and applying a modified algorithm for the residual matrix which is part of the result of the first cut; therefore a solution can be obtained without recomputing the graph and the cut from scratch. It should remarkably reduce the time but it is only worth for improvements because they must be defined over the matrix of the first segmentation. There is another possibility as we have pointed out in Section 2.3 too, and it consist of changing the Gaussian Mixture Model estimation for color distributions. Using another tool as the Improve Fast Gaussian Transform which efficiently estimates sums of Gaussians, the computational time should dramatically decreases from O(N*M) to O(N+M).

Following with the development of our method, the results of the video segmentation must provide the key information to perform the automatic audio separation. Thus it is necessary a process to link audio and video information. This is carried out according to their synchronous relation. With this aim, we apply the Audio-based video diffusion method [?] over the video signal. It provides the same sequence which is blurred where video motion is not coherent to the audio energy. Next this signal will be applied to a temporal gradient to provide an audio-visual motion map. This will be the description of the audio events into a video signal.
We have to point out, the problem we can find here is when signal presents a motion synchronous with the sound, but none audio event corresponds to that. This is a limitation we call "audio-visual motion noise" as we explained in the corresponding Sections. Its presence is difficult to avoid but we have outperformed its effect over next steps to our goal.

Working with the information about the segmentation and the last results, we can extract the amount of audio-visual motion for foreground and background. The distribution of these values have been analyzed and we can observe its division between two main levels which corresponds to the activity and inactivity of its audio track. Applying a 1D GMM with 3 components, it is possible to model the audio-visual motion behaviour and perform the detection of activity periods. One Gaussian will describe the sparse values of movement due to transitions between levels combined with the "audio-visual motion noise". The other two Gaussians will model the on/off audio states and its intersection will provide the required threshold between them. Therefore evaluating the audio-visual motion map according to this threshold we achieve the periods of activity detection for each of our regions. At this stage we have to point out a question.

If we are working with audio-visual sources which present low movements for the production of the sound, the detection of this amount of motion will be difficult to extract. Furthermore this can be masked by motion which does not correspond with the audio signal but is synchronous with the sound. Then periods of audio activity will not be successfully detected and next steps of our method will be affected.

Finally processing the audio signal according to the periods of activity for our regions, audio samples are achieved to set the prior information for the audio separation. This means, audio samples for each source can be extracted when it is known the are active alone. Therefore this result is applied as training information to estimate spectral audio models by means of GMM for our audio purposes. Assessing the audio signal with the spectral models we achieve the separation of the audio sources.
Here an important problem can appear if our sequence does not respect our audio requirements. If our sequence does not include foreground and background active alone at least in one period, then audio training samples will not be extracted. This is because the criterion to learn sources is defined for one source on and the remaining off, in order to avoid errors. The possible improvements to outperform this situation are described during our work. Therefore one option is to improve the interface and how we manage the problem to allow longer audio-visual sequences with more probabilities to include the three kind of audio visual events of the audio requirements. Another improvement will be the addition of a database with audio models which could be integrated at the interface. This will be the prior information selected by the user. Therefore it replaces the whole audio training process. This will be a new method of supervised separation of audio sources. Another point to highlight is the success of the separation will depend on the nature of the sources. If their harmonic contents presents problems when playing together, the audio separation does not achieve a successfully result. This is due to a low performance of the method which applies the spectral separation, but this is the common behaviour for the current algorithms in this framework. It none of this conditions are present we successfully perform too the last step of our method: a suitable audio separation.

To summarize we have described our algorithm with its performances, limitations and possible features to improve the method. Despite timing limitations video segmentation is successfully achieved. Provided our audio-visual sequences respect the audio requirements, audio separation gets the suitable result too. To illustrate a possible application of our method in figures 7.1, 7.2, 7.3 it is shown an audio visual composition with the results of the audio-visual extraction of sources we have develop. The audio signal it would be a mixture of the guitars.

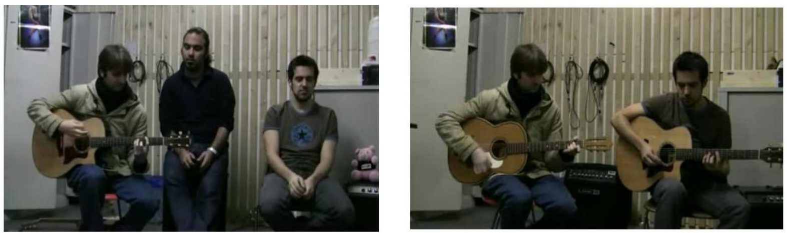


Figure 7.1: Original audio-visual sequences.

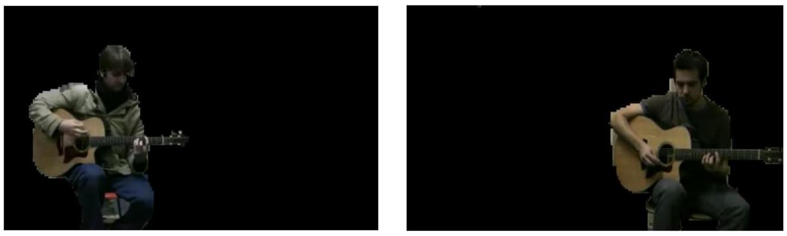


Figure 7.2: Video segmentation of two audio-visual sequences.

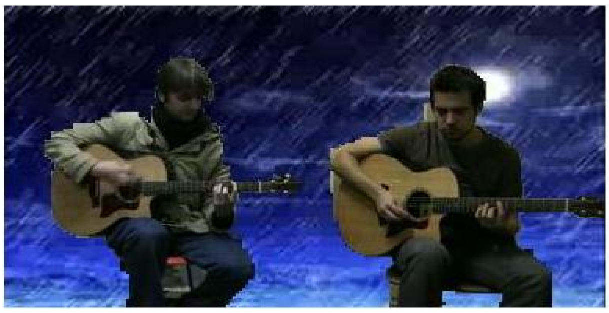


Figure 7.3: Composition of two segmented audio-visual sources into a new background.

List of Figures

List of Tables

Bibliography

[1] *Improved fast gauss transform and efficient kernel density estimation*, 2003.

[2] Xue Bai and Guillermo Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV*, pages 1–8, 2007.

[3] Zohar Barzelay and Yoav Y. Schechner. Harmony in motion. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2007.

[4] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *in ECCV*, pages 428–441, 2004.

[5] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient n-d image segmentation. *Int. J. Comput. Vision*, 70(2):109–131, 2006.

[6] Yuri Boykov and Vladimir Kolmogorov. Computing geodesics and minimal surfaces via graph cuts. *Computer Vision, IEEE International Conference on*, 1:26, 2003.

[7] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, 2001.

[8] Y.Y. Boykov and M.P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. pages I: 105–112, 2001.

[9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to algorithms, second edition, 2001.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[11] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988.

[12] Hayit Greenspan, Jacob Goldberger, and Arnaldo Mayer. Probabilistic space-time video modeling via piecewise gmm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(3):384–396, 2004.

[13] Pushmeet Kohli and Philip H. S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:2079–2088, 2007.

[14] Anna Llagostera Casanovas, Gianluca Monaci, Pierre Vandergheynst, and Rmi Gribonval. Blind Audio-Visual Source Separation based on Sparse Redundant Representations. *IEEE Transactions on Multimedia*, 2010.

[15] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23:309–314, 2004.

[16] Mark A. Ruzon and Carlo Tomasi. Alpha estimation in natural images. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1018, 2000.

[17] Jeremy G. Siek, Lie-Quan Lee, and Andrew Lumsdaine. *The Boost Graph Library: User Guide and Reference Manual (C++ In-Depth Series)*. Addison-Wesley Professional, December 2001.

[18] Christian Sigg, Bernd Fischer, Bjoern Ommer, Volker Roth, and Joachim Buhmann. Nonnegative cca for audiovisual source separation. In *IEEE Workshop on Machine Learning for Signal Processing*. IEEE Press, 2007.