TECHNISCHE UNIVERSITÄT WIEN

POLITECNICO DI MILANO

Institut für Nachrichtentechnik und Hochfrequenztechnik

INFLUENCE OF AUDIO AND VIDEO QUALITY ON SUBJECTIVE AUDIOVISUAL QUALITY

H.263 and Adaptive Multi Rate (AMR) coding

Professor: Markus RUPP

Supervisor: Michal RIES

Author: Rachele PUGLIA

Contents

Introduction

In last years the mobile application industry has proposed several metrics for video quality measurements [16], [21], [22], [23] and for audio quality measurements [12], [10]. These metrics allow to evaluate the video and audio quality perceived by people, but they assume only a single continuous medium, either audio, or video.

Nevertheless, nowadays multi-media systems are becoming more and more important. The 3G terminals supporting video streaming became reality, although the perceptual quality for such low bit rates is limited.
In UMTS packet switched streaming services (PPS) it is essential to provide required levels of customer satisfaction. We must therefore study the *subjective audiovisual quality*, so that it is possible to guarantee a good quality and to satisfy the customer expectations.

Thus, the primary importance is to find out a multi-modal model that can be used to predict audiovisual quality. Goal of our research is to estimate the quality of mobile multimedia at the user-level (perceptual quality of service) and to find optimal codecs settings for 3G streaming scenarios.

With subjective quality one means the quality estimation from user's subjective perceptual quality evaluation. It is so connected with the user level and it is often indicated with the mean opinion score (MOS) of people evaluation. The subjective quality on user level is linked with video and audio objective parameters to the application-level.

As first step we have developed some objective parameters and we have used them to design video and audio metrics that predict video and audio subjective quality (chapters 2, 3, 6). In this way we could obtain the evaluation of video-only

and audio-only quality from the physical characteristics of video sequences and audio clips.

The so obtained audio and video measurements, weighted in several ways, contribute to the subjective audiovisual quality evaluation. The contributes of the audio and video quality on the audiovisual quality can compensate each other: one can obtain the same level of audiovisual quality with several combinations of audio and video quality. The audiovisual metrics that we have designed account for this property. We have proposed several models in order to predict at the best the audiovisual quality.

To verify how much our models fitted to predict the real quality evaluation, we have performed audiovisual subjective tests using a UMTS phone and adopting the Absolute Category Rating (ACR) test method (chapter 5). We have tested typical 3G scenarios and we have encoded our clips with H.263 and MPEG-4 video codecs, and AMR and AAC audio codecs.

This work concentrates its attention particularly on the influence of AMR coding on the audiovisual quality. In the first chapter H.263 and AMR codecs are explained in their principal functions. In the last chapter (chapter 7) we have analysed the results of the quality evaluation in order to understand if AMR improves the quality. Then we have underlined advantages and disadvantages in the use of AMR and we have proposed the best bit rate combinations that allow to use the transport channel in the best way.

Chapter 1

Video and Audio Codecs

Introduction

Interest of my work is evaluating the audiovisual quality of UMTS packet-switched streaming services (PSS). But at first we can not leave out of consideration the analysis of video and audio codecs supported in PSS.
The supported video codecs are H.263 and Mpeg4. In the setting of our subjective test we have encoded test sequences with both codecs at different bit rates and we have used audio codecs AMR and AAC for speech and music coding, thereby obtaining different quality for each sequence.

The first part of the chapter concerns both mandatory and optional H.263 video codecs. With the variety of optional modes, several preferred mode combinations for operation were defined and structured into profiles of support. Moreover groupings of maximum performance parameters were determined for each profile and they are known as levels of support. Thus supported codecs standards are simply given by profile (from 1 to 8) and level (from 10 to 70). Features of admitted profiles and levels are described in paragraph 1.1.

The second part of the chapter concerns the AMR audio codec. AMR (Adaptive Multi Rate Codec) was designed to achieve an improved standard of voice quality. It attains this goal by dynamically adapting its bit-rate allocation between speech and channel coding, thereby optimising speech quality in various radio channel

conditions.

1.1 Video coding and compression

Video coding allows data compression, i.e. the reduction of redundancy in data. To achieve this goal, video coding performs following steps [1]:

- color coding
- transformation
- quantization
- coding

Color Coding

It is known that human eye is more sensible to brightness changes than to chromaticity changes. Codecs make most of this human perceptual ability and give more importance to the luminance than to the chrominance. So they reduce the number of bit for chrominance coding through a chroma subsampling. We can see some example:

- from 24 bit RGB color space to 15 or 16 bit RGB
- from 24 bit RGB color space to YUV color space with 8 bit for the luminance Y, 4 bit for the chrominance Cr and 4 bit for the chrominance Cb
- from 24 bit RGB color space to 8 bit color map.

Transformation

The picture is normally divided into blocks before any transformation is made. Then, to facilitate exploiting psychovisual redundancies, each block is transformed to a domain where different frequency ranges with varying sensitivities of the

human visual system can be separated. This can be achieved by the *discrete cosine transform* (DCT).

$$F[u,v] = \frac{4C(u)C(v)}{n^2} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos\left(\frac{(2j+1)u\pi}{2n}\right) \cos\left(\frac{(2k+1)v\pi}{2n}\right) \quad (1.1)$$

where

$$C(w) = \left\{ \begin{array}{ll} \frac{1}{\sqrt{2}} & for \quad w = 0 \\ 1 \quad for & w = 1, 2, ..., n-1 \end{array} \right\}. \quad (1.2)$$

This is a two dimensional transformation. The component at null frequency (w=0) is the one that has the greatest energy. As far the components are from the null frequency, as small their energies are. The quantization makes most of this peculiarity.

Quantization

After the transformation, the numerical precision of the transform coefficients is reduced in order to decrease the number of bits in the stream. The degree of quantization applied to each coefficient is usually determined by the visibility of the resulting distortion to a human observer; high-frequency coefficients can be more coarsely quantized than low-frequency coefficients, for example. Quantization is the stage that is responsible for the lossy part of compression.

Coding

Before coding, the quantized coefficients are taken from the most important one in a "zig-zag" way. Then a Variable Length Coding (VLC) is often used by exploiting the redundancy between the quantized coefficients in the bitstream. VLC relies on the fact that certain symbols occur much more frequently than others. The most frequent symbols are so encoded with a lower number of bits (the most frequent one is codified with only one bit). One of the most popular VLC schemes is Huffman coding.

1.1.1 Inter-frame coding

A key aspect of digital video compression is exploiting the similarity between successive frames in a sequence instead of coding each picture separately.

A simple method for temporal compression is frame differencing, where only the pixel-wise differences between successive frames are coded. In this way the frame so calculated from the previous frame is predicted (P-frame, Figure 1.1).

$$I(x, y, k) = I(x + dx, y + dy, k + 1) - I(x + dx, y + dy, k - 1) \tag{1.3}$$

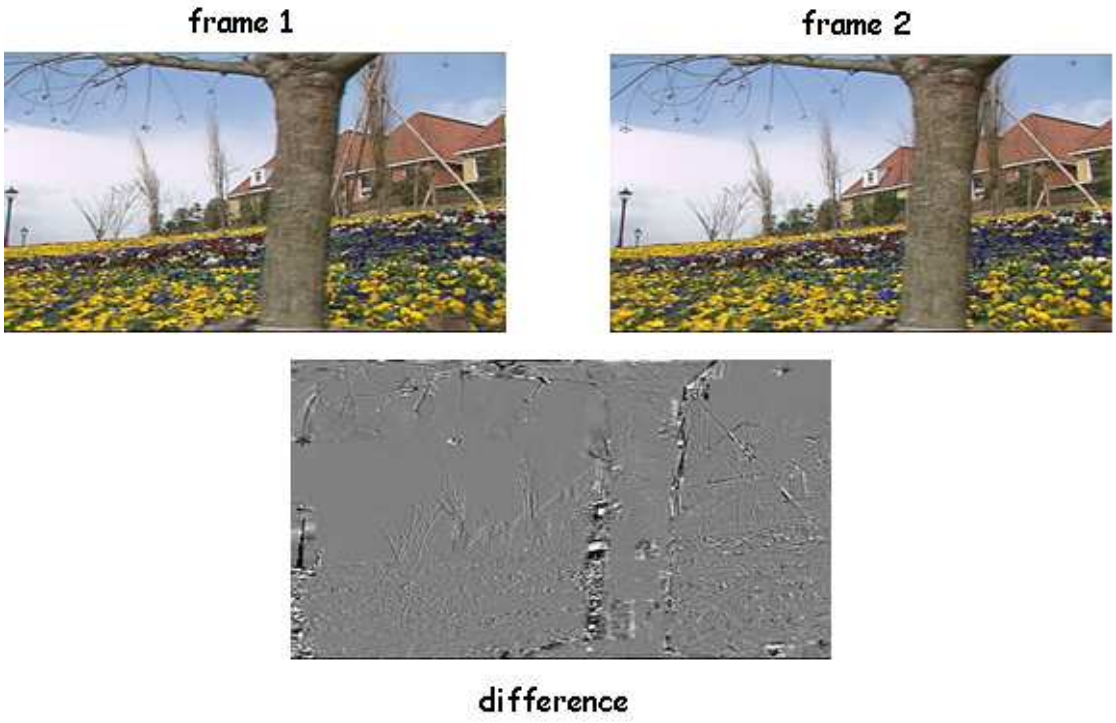

Figure 1.1: example of a difference between 2 consecutive frame.

Higher compression can be achieved using motion estimation, a technique for describing a frame based on the content of nearby frames with the help of motion vectors. By compensating for the movements of objects in this manner, the differences between frames can be further reduced. This method works dividing each frame in macro blocks. For every block it searches around the block in the previous frame for a better matching block and encodes position and error difference. The coded block position is called motion vector.

The codec can search the same block both in previous and in future frame. In this case the so called B-frame (bidirectional predicted frame) is predicted from both sides (Figure 1.2).

$$I(x, y, k) = a - I(x + dx, y + dy, k - 1) + a + I(x + dx, y + dy, k + 1) \quad (1.4)$$

The I-frames is not predicted frame.

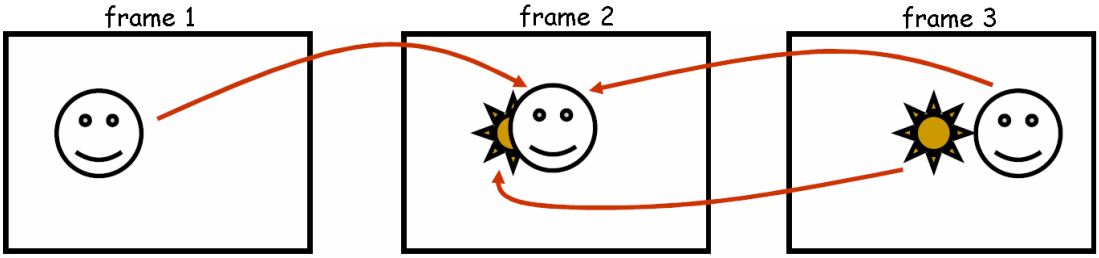

Figure 1.2: bidirectional predicted frame is predicted from both sides previous and future frames.

After the data compression the bit rate is:

$$\left(\frac{width \cdot heigth \cdot depth \cdot FR}{compressionfactor}\right) = BR \quad (1.5)$$

1.2 H.263 Video Codec

In H.263 there are different optional modes for operation. Through profiles and level parameters one can simply describe the capabilities of a coder. Several preferred mode combinations for operation are defined and structured into *"profiles" of support*. Moreover, some groupings of maximum performance parameters are defined as *levels of support* for these profiles.

3GPP technical specifications define PSS protocols and codecs [2]. The mandatory video codec is **H.263** profile 0 level 10. The codec H.263 profile 3 level 10 is instead optional [2].

The **profile 0** is the Baseline Profile, with no optional modes of operation. This means that no motion vectors are provided and that custom picture formats

(CPFMT) and custom picture clock frequencies (CPCFC) are supported [3]. Encoders and decoders supporting custom picture formats and/or custom picture clock frequencies are recommended to follow these rules:

1. A decoder for any profile and level defined herein that supports a maximum picture format shall support all standard picture formats smaller or equal in both height and width than those of the maximum supported picture format. For example, a decoder supporting a custom picture format of 720×288 shall support CIF, QCIF and sub-QCIF picture decoding.

2. A decoder for any profile and level defined herein that supports custom picture formats shall support all standard or custom picture formats having both height and width smaller than or equal to those of the maximum supported picture format.

3. A decoder for any profile and level defined herein that supports a minimum picture interval with the standard picture clock frequency of (30 000)/1001 units per second shall support the same or smaller minimum picture interval for all supported picture formats having both height and width smaller than or equal to those of the maximum picture format at which the minimum picture interval is specified.

4. A decoder for any profile and level defined herein that supports a minimum picture interval and supports custom picture clock frequencies shall support the use of any picture clock frequency with the same or larger picture interval for all supported picture formats having both height and width smaller than or equal to those of the maximum picture format at which the minimum picture interval is specified.

The Version 2 Interactive and Streaming Wireless Profile, designated as **Profile 3**, is composed of the baseline design plus the following modes [3]:

1. Advanced INTRA Coding- Use of this mode improves the coding efficiency for INTRA macroblocks (whether within INTRA pictures or predictively-coded pictures). The additional computational requirements of this mode are minimal at both the encoder and decoder (as low as a maximum of 8 additions/subtractions per 8×8 block in the decoding process plus the use of a different but very similar VLC table in order to obtain a significant improvement in coding efficiency).

2. Deblocking Filter - Because of the significant subjective quality improvement that may be realized with a deblocking filter, these filters are widely in use as a method of post-processing in video communication terminals. As with the Advanced Prediction mode, this mode also includes the four-motion-vector per macroblock feature and picture boundary extrapolation for motion compensation, both of which can further improve coding efficiency. The computational requirements of the deblocking filter are several hundred operations per coded macroblock, but memory accesses and computational dependencies are uncomplicated. This last point is what makes the Deblocking Filter preferable to Advanced Prediction for some implementations. Also, the benefits of Advanced Prediction are not as substantial when the Deblocking Filter is used as well. Thus, the Deblocking Filter is included in this basic package of support.

3. Slice Structured Mode - The Slice Structured mode provides resynchronization points within the video bitstream for recovery from erroneous or lost data.

4. Modified Quantization - This mode includes an extended DCT coefficient range, modified DQUANT syntax, and a modified step size for chrominance. The first two features allow for more flexibility at the encoder and may actually decrease the encoder's computational load (by eliminating the need re-encode macroblocks when coefficient level saturation occurs). The third feature noticeably improves chrominance fidelity, typically with little added

bit-rate cost. At the decoder, the only significant added computational burden is the ability to parse some new bitstream symbols.

Both the profile 0 and the profile 3 are provided in PSS services as **level 10** [2].
In fact for each profile it is defined a level of support, that is a group of parameters which gives maximum performance in a profile. Seven levels of performance capability are defined (from level 10 to 70). Level 10 is the minimum level of performance capability. It supports QCIF and sub-QCIF resolution and it is capable of operations with a bit rate up to 64000 bits per second with a picture decoding rate up to (15 000)/1001 pictures per second.
Upper levels support greater resolution and bit rate [3].

1.3 AMR Audio Codec

Adaptive Multi Rate (AMR) is designed as the fourth speech codec defined for the GSM system and then it has been used also in the UMTS technology. Its goal was achieving an improved standard of voice quality and greater capacity. It reaches this goal and does not require any additional hardware investment.

Unlike previous GSM speech codecs which operate at a fixed rate and constant error protection level, the AMR speech codec adapts its error protection level to the local radio channel and traffic conditions. In error prone environments that produce a high amount of errors, more bits are used for error correction to obtain robust coding. However, when transmission conditions are good, fewer bits are needed for sufficient error protection and more can therefore be allocated for speech coding [4]. This is shown in the figure Figure 1.3.

AMR generates improved speech quality in both half-rate (bit rate of 22.8 Kbps) and full-rate modes (bit rate of 11.4 Kbps) by varying the balance between speech and channel coding for the same gross bit-rate. The codec AMR can operate at different bit rates from 4.75 up to 12.2 Kbps. In UMTS the AMR bit rates are

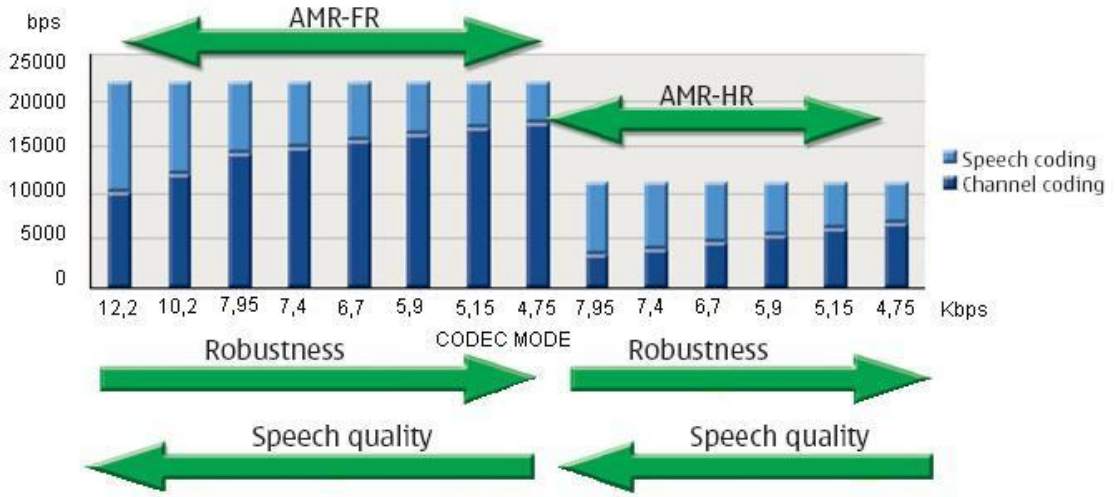

Figure 1.3: Dynamic bit-rate allocation between speech and channel coding.

controlled by the radio access network and do not depend on the speech activity. For example when in the full rate mode the codec mode is 4.75 Kbps, one can notice that this part of bit rate is used for speech coded bits and the remaining 18.05 kbps are used for channel coded bits. So in this case the robustness of the channel is very high, contrary to the speech quality.

This process, known as codec mode adaptation, results in improved voice quality throughout the cell and increases overall coverage, but it is especially noticeable at cell edges and deep inside buildings.

The codec mode adaptation allows also a higher resilience to errors and hence to interference. This can be used to increase capacity by operating a tighter frequency reuse pattern.

General overview of the speech processing functions

The AMR speech coder consists of the multi-rate speech coder, a source controlled rate scheme including a voice activity detector and a comfort noise generation system, and an error concealment mechanism to combat the effects of transmission

errors and lost packets [5].

The *multi-rate speech coder* is a single integrated speech codec with eight source rates from 4.75 kbps to 12.2 kbps, and a low rate background noise encoding mode. The speech coder is capable of switching its bit-rate every 20 ms speech frame (160 speech samples) upon command.

A reference configuration where the various speech processing functions are identified is given in Figure 1.4. In this figure, the relevant specifications for each function are also indicated.

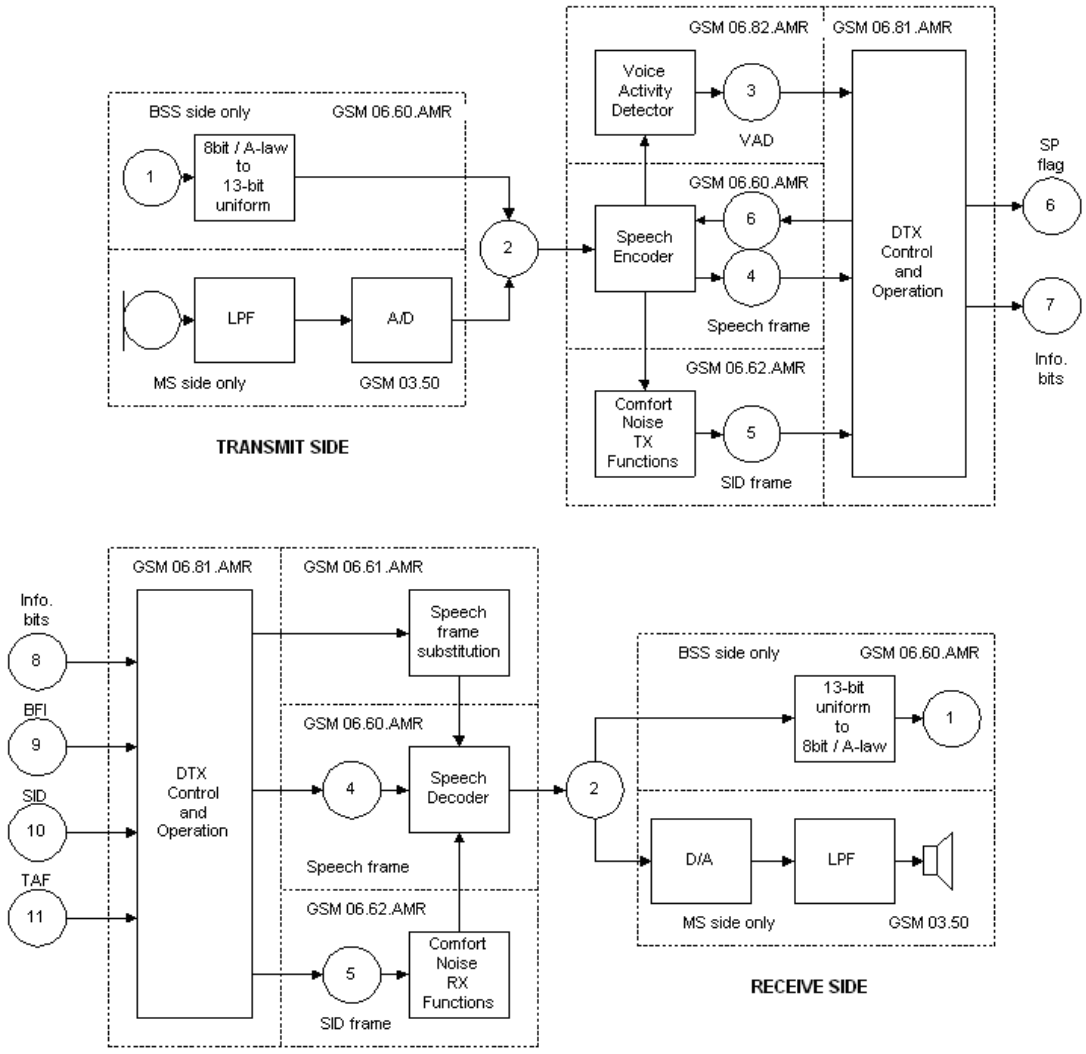

Figure 1.4: Overview of audio processing functions.

1. *8-bit* A*-law or* μ*-law PCM, 8 000 samples/s;*

2. *13-bit uniform PCM, 8 000 samples/s;*
3. *Voice Activity Detector (VAD) flag;*
4. *Encoded speech frame, 50 frames/s, number of bits/frame depending on the AMR codec mode;*
5. *Silence Descriptor (SID) frame;*
6. *TX-type, 2 bits, indicates whether information bits are available and if they are speech or SID information;*
7. *Information bits delivered to the 3G AN;*
8. *Information bits received from the 3G AN;*
9. *RX-type, the type of frame received quantized into three bits.*

As shown in the figure, the speech encoder takes its input as a 13-bit uniform Pulse Code Modulated (PCM) signal via an 8-bit A-law or μ-law to 13-bit uniform PCM conversion. The encoded speech at the output of the speech encoder is packetized and delivered to the network interface. In the receive direction, the inverse operations take place. The detailed mapping between input blocks of 160 speech samples in 13-bit uniform PCM format to encoded blocks (in which the number of bits depends on the presently used codec mode) and from these to output blocks of 160 reconstructed speech samples is described in [6].

During a normal telephone conversation, the participants alternate so that, on the average, each direction of transmission is occupied about half of the time. *Source controlled rate (SCR)* is a mode of operation where the speech encoder encodes speech frames containing only background noise with a lower bit-rate than normally used for encoding speech.
The following functions are required for the source controlled rate operation:

- a *Voice Activity Detector (VAD)* on the TX side: the input to the VAD is the input speech itself together with a set of parameters computed by the

adaptive multi-rate speech encoder. The VAD uses this information to decide whether each 20 ms speech coder frame contains speech or not.

- evaluation of the background acoustic noise on the TX side, in order to transmit characteristic parameters to the RX side.

- *generation of comfort noise* on the RX side during periods when no normal speech frames are received.The synthesis of this artificial noise is based on the received non-speech parameters.

An other important mechanism of AMR is the *error concealment of lost frames.* In order to mask the effect of isolated lost frames, the speech decoder shall be informed and the error concealment actions shall be initiated, whereby a set of predicted parameters are used in the speech synthesis.

Chapter 2

Video objective parameters

Introduction

Video quality assessment has become rather well established by now, as evidence by the number of research publication available. We have chosen to envelope four objective parameters that come from the ANSI standard [16]. They are connected with gain and loss of intensity and orientation of the spatial activity, and consider how the codec operates. In ANSI standard they are indicated as *sigain*, *siloss*, *hvgain* and *hvloss*.

The parameter sigain depends on the standard deviation of the luminance. If the codec operates through an edge sharpening, the standard deviation of the luminance increases. The parameter indicates this improvement of the quality. Siloss concerns instead the loss of the spatial activity when a blurring phenomenon happens.

The parameters hvgain and hvloss investigate the orientation of the spatial activity. Hvgain indicates if a blockiness phenomenon happens, i.e. if the ratio of horizontal and vertical edge to diagonal edge increases in the processed sequence. The hvloss parameter shows if the horizontal and vertical edges suffer more blurring than the diagonal ones.

In the following paragraphs it is explained how these parameters are implemented.

2.1 General Model

This standard specifies a method for estimating the video performance of a one-way video transmission , as shown in Figure 2.1. The objective video performance estimator is defined for the end-to-end transmission quality between the two points shown in figure. The encoder can utilize various compression methods. We have used MPEG-4 and H.263. Nevertheless it is assumed that any digital transport involved is error free.

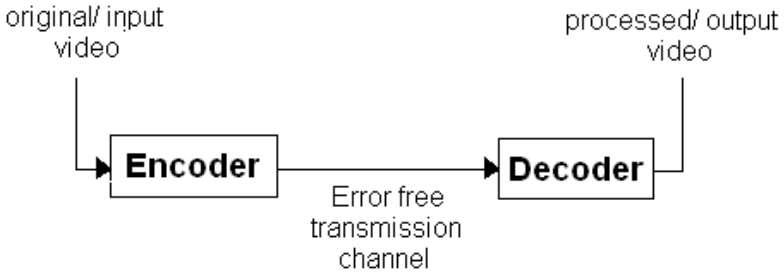

Figure 2.1: Reference model for measurement of one-way video transmission system performance.

This metric is based on four different quality parameters that measure the perceptual effects of a wide range of impairments such as blurring, block distortion, unnatural motion, noise and error blocks. Each quality parameter is calculated through a quality feature, defined as a quantity of information associated with a spatial-temporal sub-region of a video stream.

2.1.1 Quality Features

This section describes a set of quality features that characterize perceptual changes in the spatial and temporal properties of video streams. Normally, a perceptual filter is applied to the video stream to enhance some property of perceived video quality, such as edge information. After this perceptual filtering, features are extracted from spatial-temporal sub-regions using a mathematical function. Finally, a perceptibility threshold is applied to the extracted features.

Our two features (f_{SI13} and f_{HV13}) derive from spatial gradients and are used to characterize perceptual distortions of edges.

As first step, two edge enhancement filters are applied to the original and processed Y(luminance) video frames. These filters have size 13 x 13 and are applied separately. They increase spatial gradients in the horizontal (H) and vertical (V) direction respectively. For the numeric values one can see the matlab program in appendix.

Then the spatial-temporal regions (S-T regions) are defined. Each S-T region describes a block of pixels. Its sizes are described by the number of pixel horizontally, the number of frame lines vertically and the time duration of the regions, given in units of equivalent video frames. In our metric the S-T region spans 8 horizontal pixel x 8 vertical lines x 5 video frames. After the video stream has been divided into S-T regions, the temporal axis of the feature (t) no longer corresponds to individual frames, but each instant corresponds to five frames (Figure 2.2).

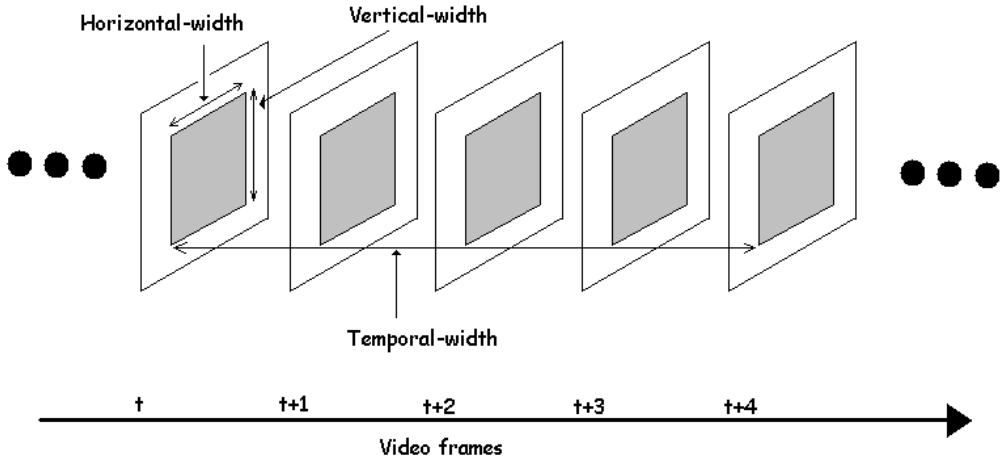

Figure 2.2: Example spatial-temporal (S-T) region size for extracting features.

For each S-T region the H and V filter responses , denoted as $H(i,j,t)$ and $V(i,j,t)$, are converted into polar coordinates ($R(i,j,t)$, θ) using the relationship:

$$R(i,j,t) = \sqrt{H(i,j,t)^2 + V(i,j,t)^2} \tag{2.1}$$

and

$$\theta(i,j,t) = \tan^{-1}\left(\frac{V(i,j,t)}{H(i,j,t)}\right). \tag{2.2}$$

The first feature, denoted as f_{SI13}, is computed simply as the standard deviation (std) over the S-T region of the $R(i,j,t)$ samples, and then clipped at the perceptibility threshold of P.

$$f_{SI13} = \{std[R(i,j,t)]\}|_P : i,j,t \in \{S-Tregion\} \tag{2.3}$$

This feature is sensitive to changes in the overall amount of spatial activity within a given S-T region.

The second feature is sensitive to changes in angular distribution, or orientation, of spatial activity. Complementary images are computed with the shaded spatial gradient distributions shown in Figure 2.3.

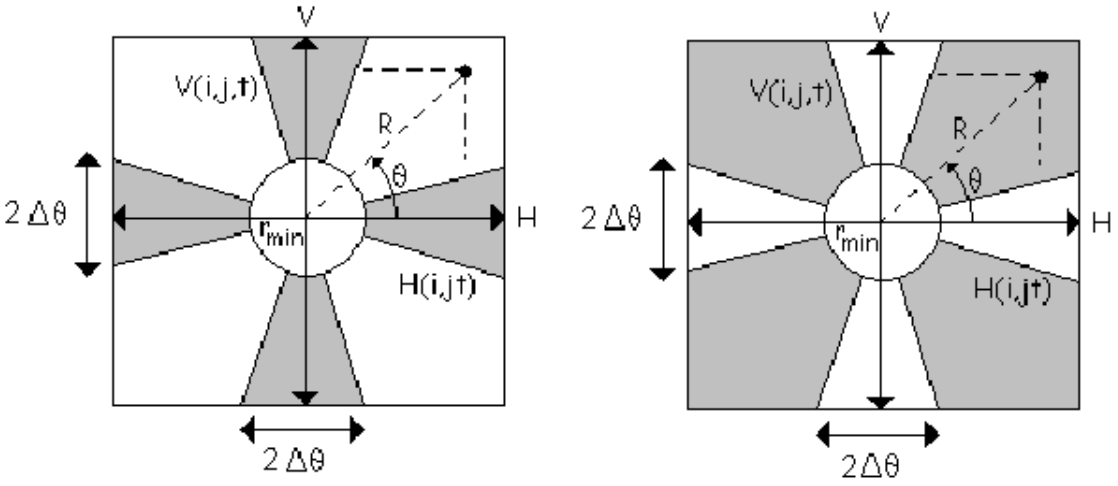

Figure 2.3: Division of horizontal(H) and vertical(V) spatial activity into HV (left) and $\overline{HV}$ (right) distributions.

The image with horizontal and vertical gradients, denoted as HV, contains the $R(i,j,t)$ pixels that are horizontal and vertical edges (pixels that are diagonal edges are zeroed). The image with diagonal gradients, denoted as $\overline{HV}$, contains the $R(i,j,t)$ pixels that are diagonal edges (pixels that are horizontal and vertical

edges are zeroed). Pixels in HV and $\overline{HV}$ can be represented mathematically as:

$$HV(i,j,t) = \begin{cases} R(i,j,t) \\ \text{if } R(i,j,t) \geq r_{min} \text{ and } m\frac{\pi}{2} - \Delta\theta < \theta(i,j,t) < m\frac{\pi}{2} + \Delta\theta \ (m\text{=}0,1,2,3) \\ \\ 0 \\ otherwise \end{cases} \tag{2.4}$$

and:

$$\overline{HV}(i,j,t) = \begin{cases} R(i,j,t) \\ \text{if } R(i,j,t) \geq r_{min} \text{ and } m\frac{\pi}{2} + \Delta\theta < \theta(i,j,t) < (m+1)\frac{\pi}{2} - \Delta\theta \\ (m = 0,1,2,3) \\ \\ 0 \\ otherwise \end{cases} \tag{2.5}$$

where: $i,j,t \in \{S-Tregion\}$.

For the computation of HV and $\overline{HV}$, the values recommended by [16] for r_{min} and $\Delta\theta$ are 20 and 0.225 radians respectively. Feature f_{HV13} for one S-T region is then given by the ratio of the mean of HV to the mean of $\overline{HV}$, where these resultant means are clipped at their perceptibility threshold P, namely:

$$f_{HV13} = \frac{\{mean[HV(i,j,t)]\}|_P}{\{mean[\overline{HV}(i,j,t)]\}|_P}. \tag{2.6}$$

The recommended perceptibility threshold P for the mean of HV and $\overline{HV}$ is 3. The f_{HV13} feature is sensitive to changes in the angular distribution of spatial activity within a given S-T region.

As each feature is characteristic of one S-T region, we compute as many f_{SI13} and f_{HV13} features as the number of regions in the video stream. For further information one can see the feature function in matlab implementation in appendix.

2.1.2 Quality Parameters

Quality parameters that measure distortions in the video quality due to gains and losses in the feature values are first calculated for each S-T region by comparing the original feature values, $f_o(s,t)$, with the corresponding processed feature values, $f_p(s,t)$.

Loss and gain are normally examined separately, since they produce fundamentally different effects on quality perception (e.g. loss of spatial activity due to blurring and gain of spatial activity due to noise or blocking). For this aim we have utilized the two following functions:

$$ratioloss(s,t) = np\left[\frac{f_p(s,t) - f_o(s,t)}{f_o(s,t)}\right], \tag{2.7}$$

and

$$loggain(s,t) = pp\left[\log_{10}\left(\frac{f_p(s,t)}{f_o(s,t)}\right)\right], \tag{2.8}$$

where *pp* is the positive part operator (negative values are replaced with zero), and *np* is the negative part operator (positive values are replaced with zero).

These visual masking functions imply that impairment perception is inversely proportional to the amount of localized spatial or temporal activity that is present. In other words, spatial impairments become less visible as the spatial activity increases, and temporal impairments become less visible as the temporal activity increases.

At first we have computed the *ratioloss function* on the f_{HV13} feature and on the f_{SI13} one, where the threshold p in Equation 2.3 is 12. We denote these two set of parameters (matrixes) as *hvloss** and *siloss**, respectively. Then we have calculated the *loggain function* on the f_{HV13} feature and on the f_{SI13} one, where the threshold p in Equation 2.6 is 8. We denote these two set of parameters (matrixes) as *hvgain** and *sigain**, respectively. For further information one can see the region function in matlab implementation in appendix.

The hvloss*, siloss*, hvgain* and sigain* matrixes give both spatial information (given by the rows) and temporal information (given by the columns). Now,

impairments from the S-T region with the same time index t are pooled using a *spatial collapsing function*. Spatial collapsing yields a time history of parameters values.

The spatial function *below5%* is applied on hvloss* and siloss*. For each temporal index t (i.e. for each column of the matrixes), this function sorts the parameter values from low to high. Then it computes the average of all the parameter values that are less than or equal to the 5% threshold level. For loss parameters, it produces a parameter that is indicative of the worst quality over space.

The spatial function *above95%* is instead applied on hvgain*. For each temporal index t (i.e. for each column of the matrix), this function sorts the parameter values from low to high. Then it computes the average of all the parameter values that are greater than or equal to the 95% threshold level. For gain parameters, it produces a parameter that is indicative of the worst quality over space.

At last the spatial function *mean* is applied on sigain*. For each temporal index t(i.e. for each column of the matrix), this function computes the average of all the parameter values. It produces a parameter that is indicative of the average quality over space.

After that the spatial collapsing functions have been applied, our four parameters have become row vectors. Now all the elements of each vector are pooled using a *temporal collapsing function* to produce an objective parameter for the video clip.

The temporal function *10%* is applied on the siloss* row vector. It sorts the time history of the parameter values from low to high and select the 10% threshold level. This is the final ***siloss parameter***. For loss parameters, the temporal function produces a parameter that is indicative of the worst quality over time. For gain parameters, it produces a parameter that is indicative of the best quality over time.

The temporal function *mean* is instead applied on the sigain*, hvoss* and hvgain* row vectors. The three gotten parameters are indicative of the average

quality over time. The ***hvgain parameter*** is so defined completely.

On sigain* parameter and on raised to two hvloss* parameter, a clipping function is calculated to reduce the sensitivity of the parameters to small impairments. A clipping function is mathematically represented as:

$$clip_T = \begin{cases} max(p,T) - T & \text{if p is all positive} \\ min(p,T) - T & \text{if p is all negative} \end{cases} \tag{2.9}$$

where p is the parameter and T the threshold. The threshold T is 0.004 for the sigain* parameter and 0.06 for the squared hvloss*. After the application of this clip function, also the ***sigain*** and ***hvloss parameters*** are defined completely.

Chapter 3

Audio objective parameters

Introduction

Digital speech encoding and transmission involve a four-way compromise between complexity, delay, bit rate, and the perceived quality of decoded speech. Complexity, delay, and bit rate can often be quantified in fairly straightforward ways, but perceived quality can be more difficult to measure. Subjective listening or conversation tests can be used to gather firsthand evidence about perceived speech quality, but such tests are often fairly expensive, time-consuming, and labor-intensive. For this reason, although the most important measurements of perceived speech quality rely always on formal subjective tests, numerous attempts have been made to supplement them with *objective estimators* of perceived speech quality.
For example the parameters Signal-to-noise ratio (SNR) or segmental SNR (SNRseg) can provide indications of perceived quality in some waveform-preserving speech systems. Unfortunately, when they are used to evaluate more general coding and transmission systems, SNR and SNRseg often show little, if any, correlation to perceived speech quality [10]. So, the objective estimation of the perceived quality of highly compressed digital speech, possibly with bit errors or frame erasures has remained an open question.

Researchers have recently attempted a *perception-based approach* to create estimators that hear speech signals through the same transformations that humans

hear them.

The most important of these estimators are the parameters *PESQ (Perceptual Evaluation of Speech Quality)* [12] and *AD (Auditory Distance)* [11].

In the first part of this chapter we analyse in detail the metric that allows to build the objective parameter AD. This new objective estimation approach uses a simple but effective perceptual transformation obtained through the use of a Bark frequency scale, and a distance measure that consists of a hierarchy of measuring normalizing blocks. Each measuring normalizing block integrates two perceptually transformed signals over some time or frequency interval to determine the average difference across that interval. This difference is then normalized out of one signal, and is further processed to generate one or more measurements. The linear combination of these measurements gives the Auditory Distance.

The second part of the chapter concentrates its attention on the parameter PESQ. It compares an original signal X(t) with a degraded signal Y(t) that is the result of passing X(t) through a communications system. In the first step of PESQ a series of delays between original input and degraded output are computed. Then, based on the set of delays that are found, PESQ compares the original (input) signal with the aligned degraded output of the device under test using a perceptual model. The key to this process is transformation of both the original and degraded signals to an internal representation that is analogous to the psychophysical representation of audio signals in the human auditory system, taking account of perceptual frequency and loudness. Through this process two parameters are found and combined to give an objective audio MOS (mean opinion square).

3.1 Auditory Distance

The main steps in the metric of AD are the *delay estimation* between the two input vectors of speech samples, the *perceptual transformation* and the *distance measure* (Figure 3.1).

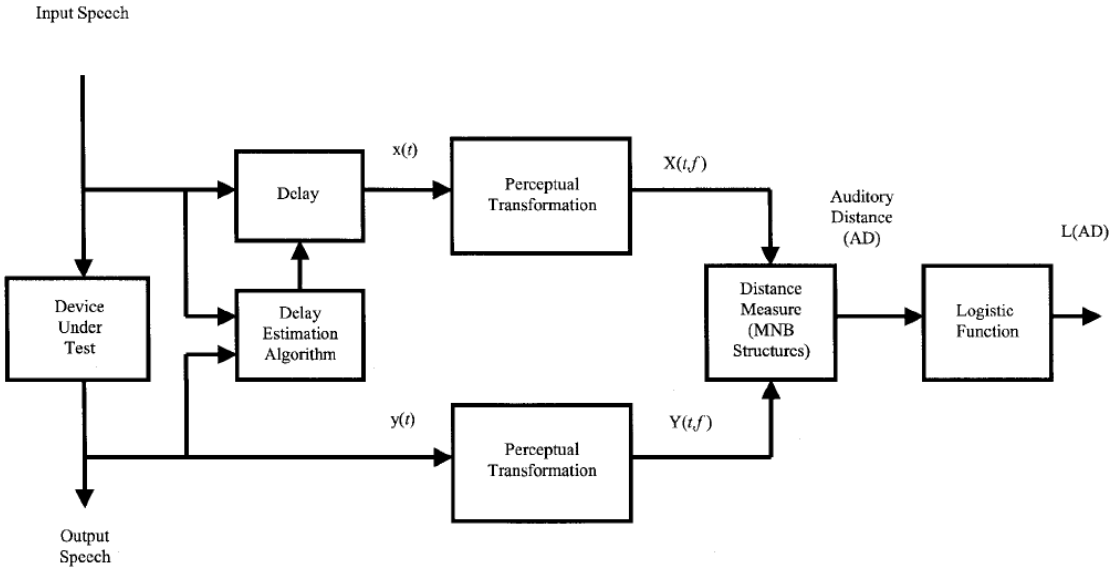

Figure 3.1: High-level block diagram of the objective estimation approach.

Delay Estimation

In the following steps to develop the audio parameter AD, the two input vectors containing the samples of the original and of the degraded signals, must have the same length and be synchronized.

In order to obtain the synchronization we have developed an easy algorithm. Both in the case of AMR codec and in that one of AAC codec, we have calculated the cross-correlation between the original and the degraded signals. The point in which the maximum of the cross-correlation appears, indicates the delay of the degraded signal towards the original one. It follows the synchronization of the two signals (see Matlab implementation in appendix).

If the utilized audio codec is AMR, the original vector x is longer than the degraded one (y). The samples at the end of the vector x are eliminated so that the length of x and y becomes the same. If the utilized audio codec is instead AAC, the vector y is the longer one. So it is cut in order to have the same length as x.

Perceptual Transformations

The goal of *perceptual transformation* is to mimic human hearing so that only information that is perceptually relevant is retained. It is known that the ear's frequency resolution is not uniform on the Hertz scale and that perceived loudness is related to signal intensity in a nonlinear way. These are the most important properties to model.

Researchers have arrived at a very simple yet effective perceptual transformation that is built from a sequence of already established steps. This perceptual transformation is applied to frequency domain representations of the speech signals. Speech signals are broken into frames, multiplied by a Hamming window, and then transformed to the frequency domain using an FFT. Our implementation is based on a sample rate of 8000 samples/s, a frame size of 128 samples (16 ms) and a 50 per cent frame overlap.

The nonuniform frequency resolution of the ear is treated by the use of a *psychoacoustic frequency scale*: a Bark frequency scale. The Hertz scale frequency variable f is replaced with the Bark frequency scale variable b using the relationship

$$b = 6 \cdot \sinh^{-1}\left(\frac{f}{600}\right) \tag{3.1}$$

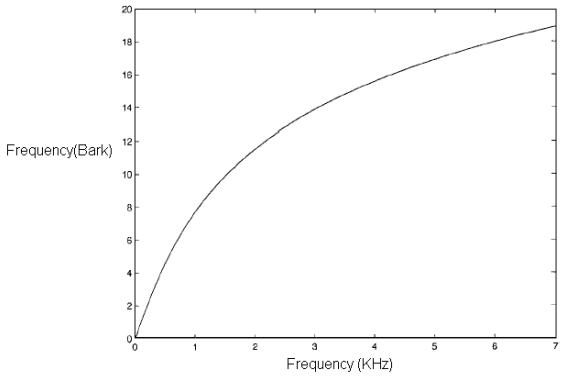

Figure 3.2: Hertz to Bark transformation.

This relationship is plotted in Figure 3.2. Note that b increases approximately linearly with f below about 500 Hz, and b increases according to a compressive nonlinearity above about 500 Hz.

Many models for loudness perception as a function of signal intensity are available. We have chosen to use a logarithm to convert signal intensity to *perceived loudness*.

Distance Measures

The measure of the *distance* between the two perceptual transformed signals happens through a hierarchical structure of Measuring Normalizing Blocks (MNB).

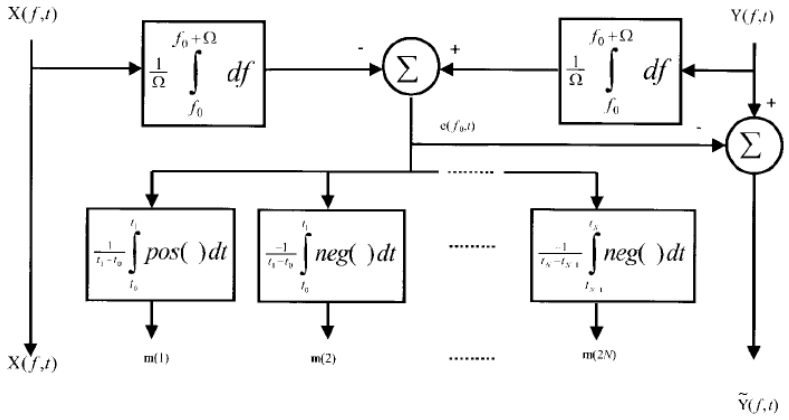

Figure 3.3: Time measuring normalizing block (TMNB).

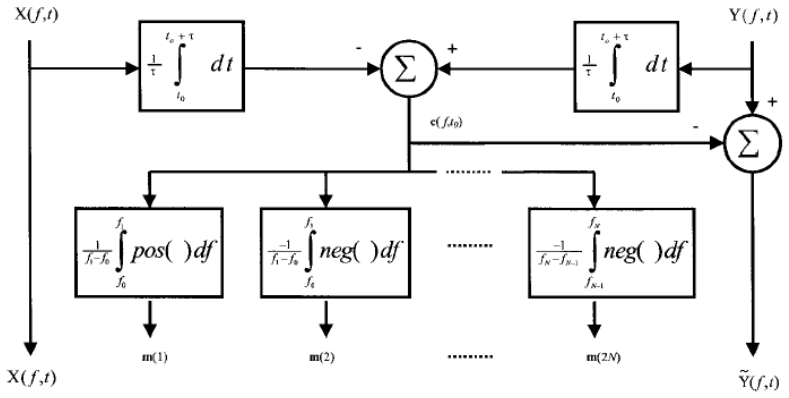

Figure 3.4: Frequency measuring normalizing block (FMNB).

A time measuring normalizing block (TMNB) is shown in Figure 3.3 and a frequency measuring normalizing block (FMNB) is given in Figure 3.4. Each of these blocks takes perceptually transformed input and output signals (X(f, t) and

Y(f, t), respectively) as inputs, and returns a set of measurements.
The TMNB integrates over some frequency scale the original and the degraded signals, then measures differences between the so integrated degraded and original signals, and normalizes the output signal at multiple times.
An FMNB integrates over some time scale the original and the degraded signals, then measures differences between the so integrated degraded and original signals, and normalizes the output signal at multiple frequencies.

We hypothesized that hierarchical structures that work from larger time and frequency scales down to smaller time and frequency scales would be most likely to emulate listeners' patterns of adaptation and reaction to spectral differences. There are two MNB structures that offer relatively low complexity and high performance as estimators of perceived speech quality. These structure are shown in Figure 3.5 and Figure 3.6. In our program we have elected to implement the structure 1 (Figure 3.5).

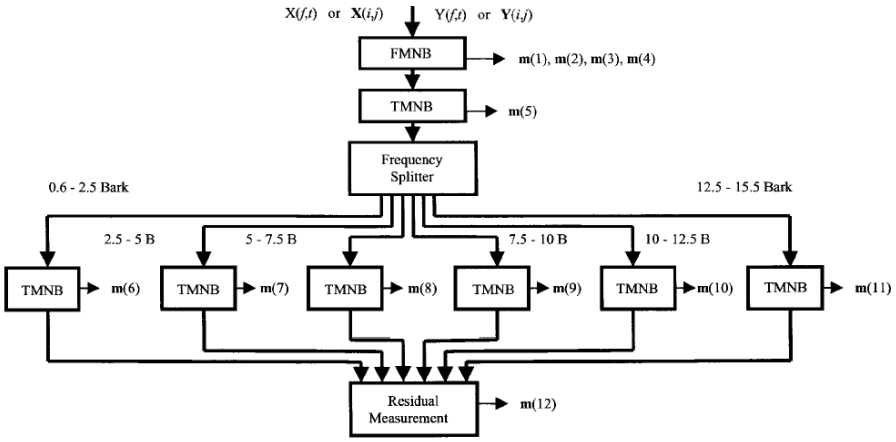

Figure 3.5: MNB structure 1.

Both MNB structures start with an FMNB that is applied to the input and output signals at the longest available time scale. Four measurements are extracted and stored in the measurement vector m. These measurements cover the lower and upper band edges of telephone band speech (from 0 to 500 and from 3000 to 3500 Hz). In MNB structure 1, a TMNB is then applied to the input and output signals at the largest frequency scale (approximately 15 Bark). Finally a residual

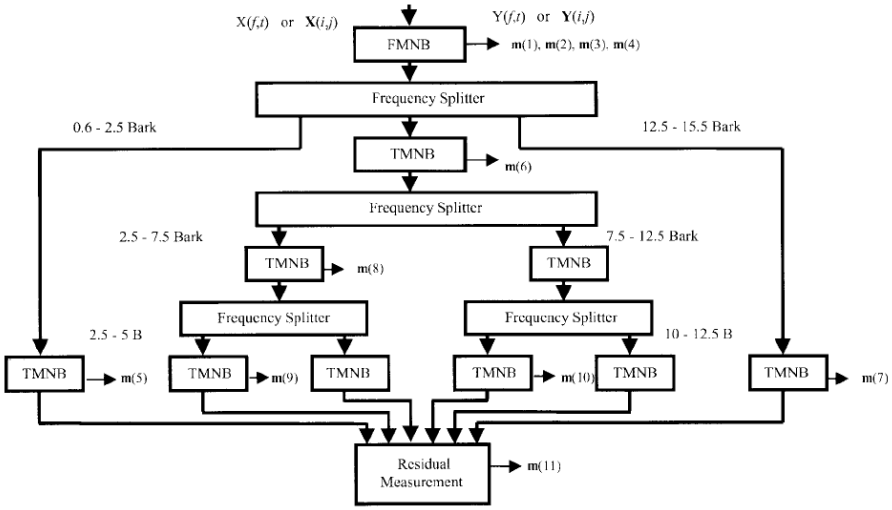

Figure 3.6: MNB structure 2.

measurement is made.

We can loosely describe the action of these MNB structures as a dynamic decomposition of a codec output signal. This decomposition proceeds in a space that is defined partly by human hearing and judgment (through the MNB structure) and partly by the codec input signal. The parameters of this dynamic decomposition are combined linearly to form a measure of the perceptual distance between two speech signals. Each measure m taken at different frequencies, is weighed with different coefficients, because the human's ears do not perceive all the frequencies in the same way. The set of coefficients m is given in appendix with the Matlab implementation. The value that results from this linear combination is called ***auditory distance (AD)***:

$$AD = \bar{w}^T \cdot \bar{m} \tag{3.2}$$

where w is a length 12 (MNB structure 1) or 11 (MNB structure 2) vector of weights. In practice, AD values are nonnegative. When the input and output signals are identical, all measurements are zero and AD is zero. As the input and output signals move apart perceptually, AD increases.

Now we want to use the generated AD distance values to estimate perceived speech quality. In our work we have performed ACR tests (Absolute Category

Rating test, see chapter 5) where the mean opinion score (MOS) scale in the interval from 1 to 5 is used. Therefore we must map AD values into a finite range. For this reason we select the following logistic function with asymptotes at 0 and 1:

$$L(AD) = \frac{1}{(1 + e^{AD-4.6877})} \tag{3.3}$$

$L(AD)$ is a decreasing function of AD. The function makes a compression at the high and low quality extremes, and it is nearly linear over the intermediate quality range.

The here described metric generates very useful estimates of perceived speech quality. On the other hand, we do not claim any direct, firm, correspondence between the algorithmic steps given above and the process of human audition and judgment. That is, the MNB structures are able to emulate the responses of listeners, but they do not directly explain or explicitly model how listeners arrive at those responses.

The implementation of AD metric is enclosed in appendix. In this implementation we have followed the instructions in reference [11].

3.2 Perceptual Evaluation of Speech Quality

Perceptual Evaluation of Speech Quality (PESQ) is an objective measurement technique for estimating subjective quality in listening only tests.

Although correlations between objective and subjective scores in the benchmark were around 0.935, the PESQ algorithm cannot be used to replace subjective testing while it does not provide a comprehensive evaluation of transmission quality. Some test factors for which PESQ has demonstrated acceptable accuracy are speech input levels to a codec, transmission channel errors, packet loss, bit rates if a codec has more than one bit rate mode, environmental noise at sending side and effect of varying delay in listening only tests. So PESQ only measures the effects of one way speech distortion and noise on speech quality. The effects of loudness loss, delay in conversational tests, sidetone, talker echo, and other impairments

related to two way interaction are not reflected in the PESQ scores. Therefore, it is possible to have high PESQ scores, yet poor quality of the connection overall.

Overview of PESQ

PESQ compares an original signal *X(t)* with a degraded signal *Y(t)* that is the result of passing *X(t)* through a communications system. The output of PESQ is a prediction of the perceived quality that would be given to *Y(t)* by subjects in a subjective listening test.

At the beginning of PESQ metric a series of delays between original input and degraded output are computed. Then PESQ compares the original (input) signal with the aligned degraded output (Figure 3.7) using an internal representation of audio signals in the human auditory system, taking account of perceptual frequency (Bark) and loudness (Sone). This is achieved in several stages: time alignment, level alignment to a calibrated listening level, time frequency mapping, frequency warping, and compressive loudness scaling.

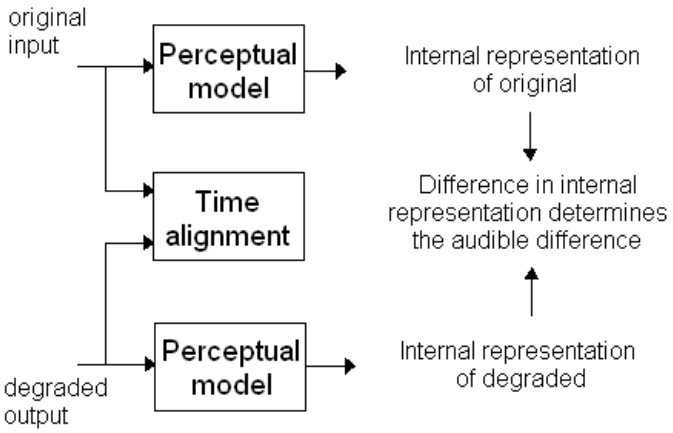

Figure 3.7: Overview of the basic philosophy used in PESQ.

The internal representation is processed to take account of effects such as local gain variations and linear filtering that may - if they are not too severe - have little perceptual significance. This is achieved by limiting the amount of compensation and making the compensation lag behind the effect. Thus minor, steady-state

differences between original and degraded are compensated. More severe effects, or rapid variations, are only partially compensated so that a residual effect remains and contributes to the overall perceptual disturbance. This allows a small number of quality indicators to be used to model all subjective effects. In PESQ, two error parameters are computed in the cognitive model; these are combined to give an objective listening quality MOS.

Pre-processing and delay estimation

It is important that test signals for use with PESQ are representative of the real signals carried by communications networks. For this reason pre-processing is often necessary to take account of filtering in the send path of a handset, and to ensure that power levels are set to an appropriate range. The gain of the system under test is not known *a priori* and may vary considerably. Thus it is necessary to level align both the original *X(t)* and degraded signal *Y(t)* to the same, constant power level. The level alignment algorithm in PESQ proceeds as follows:

- Filtered versions of the original and degraded signal are computed. The filter blocks all components under 250 Hz, is flat until 2000 Hz and then falls off with a piecewise linear response till 4000Hz, where it has a value of -500 dB.
- The average value of the squared filtered original speech samples and filtered degraded speech samples are computed.
- Different gains are calculated and applied to align both the original *X(t)* and degraded speech signal *Y(t)* to a constant target level resulting in the scaled versions $X_S(t)$ and $Y_S(t)$ of these signals.

Moreover in PESQ IRS-like receive filtered versions of the original speech signal and degraded speech signal are computed. This is implemented by a FFT over the length of the file, filtering in the frequency domain with a piecewise linear response, followed by an inverse FFT over the length of the speech file [12]. This results in

the filtered versions $X_{IRSS}(t)$ and $X_{IRSS}(t)$ of the scaled input and output signals $X_S(t)$ and $Y_S(t)$.

Now it follows the alignment routine used to determine the delay per time interval d_i(Figure 3.8).

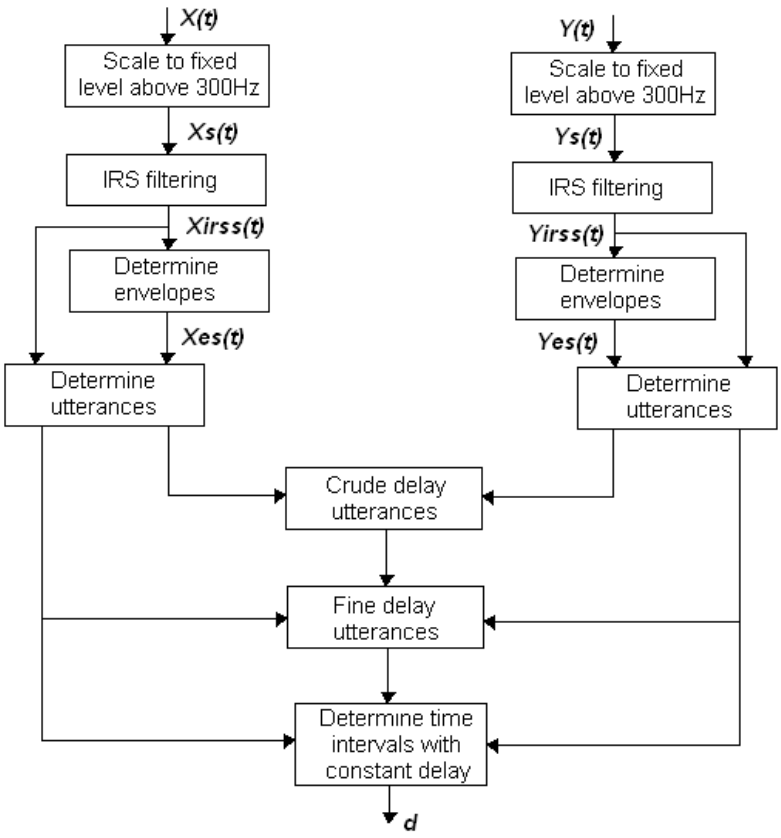

Figure 3.8: Overview of the alignment routine.

First an *envelope-based alignment* is performed. The envelopes $X_{ES}(t)_k$ and $Y_{ES}(t)_k$ are calculated from the scaled original and degraded signals $X_S(t)$ and $Y_S(t)$. The envelope is defined as $\log(\max(E(k)/Ethresh, 1))$, where *E(k)* is the energy in 4 ms frame k and *Ethresh* is the threshold of speech determined by a voice activity detector. Cross-correlation of the envelopes for the original and degraded signals is used to estimate the crude delay between them, with an approximate resolution of 4 ms.

Then a *fine time alignment* is performed. Because perceptual models are sensitive to time offsets, it is necessary to calculate a sample-accurate delay value. This is

computed as follows:

- 64 ms frames (75 per cent overlapping) are Hann windowed and cross-correlated between original and degraded signals, after the envelope-based alignment is performed.
- The maximum of the correlation, to the power 0.125, is used as a measure of the confidence of the alignment in each frame. The index of the maximum gives the delay estimate for each frame.
- A histogram of these delay estimates, weighted by the confidence measure, is calculated. The histogram is then smoothed by convolution with a symmetric triangular kernel of width 1 ms.
- The index of the maximum in the histogram, combined with the previous delay estimate, gives the final delay estimate.

Perceptual model (Figure 3.9, Figure 3.10)

The perceptual model of PESQ is used to calculate a distance between the original and degraded speech signal (PESQ score).

Input signals of the perceptual model are already scaled and filtered signals $X_{IRSS}(t)$ and $Y_{IRSS}(t)$.
The human ear performs a time-frequency transformation. In PESQ this is implemented by a short-term FFT with a window size of 32 ms. The overlap between successive time windows (frames) is 50 per cent. The power spectra - the sum of the squared real and squared imaginary parts of the complex FFT components - are stored in separate real valued arrays for the original and degraded signals. Phase information within a single Hann window is discarded in PESQ and all calculations are based on only the power representations $PX_{WIRSS}(f)_n$ and $PY_{WIRSS}(f)_n$.
The start points of the windows in the degraded signal are shifted over the delay.

The time axis of the original speech signal is left as is. If the delay increases, parts of the degraded signal are omitted from the processing, while for decreases in the delay parts are repeated.

Now the frequency axis is warped and converted to a Bark frequency scale. The Bark scale reflects that at low frequencies, the human hearing system has a finer frequency resolution than at high frequencies. This is implemented by binning FFT bands and summing the corresponding powers of the FFT bands with a normalization of the summed parts. The resulting signals are known as the *pitch power densities* $PPX_{WIRSS}(f)_n$ and $PPY_{WIRSS}(f)_n$.

Then a partial compensation of the original pitch power density is performed. This partial compensation is used because severe filtering can be disturbing to the listener. The compensation is carried out on the original signal because the degraded signal is the one that is judged by the subjects in an ACR experiment. Thus the original pitch power density $PPX_{WIRSS}(f)_n$ of each frame n is multiplied with a partial compensation factor to equalize the original to the degraded signal.
To determine this compensation factor, the power spectrum of the original and degraded pitch power densities are averaged over time. But this average is calculated only over *speech active frames* whose power is more than 1000 times the absolute hearing threshold. The compensation factor is given by the ratio of the so averaged degraded spectrum to the averaged original spectrum. The maximum compensation is never more than 20 dB. The compensated original pitch power density is indicated as $PPX'_{WIRSS}(f)_n$.

Speech active frames indicate the time interval where a real speech, without silent parts, is present. If the original and degraded speech files start or end with large silent intervals, this could influence the computation of certain average distortion values over the files. Therefore, through the comparison with a threshold value, it is estimated if each frame is silent or speech active.

After that $PPX'_{WIRSS}(f)_n$ has been performed, the distorted pitch power density is as well compensated. Short-term gain variations are partially compensated by processing the pitch power densities frame by frame. For the original and the degraded pitch power densities, the sum in each frame n of all values that exceed the absolute hearing threshold is computed. The ratio of the power in the original and the degraded files is calculated and bounded to the range $[3 \cdot 10^{-4}, 5]$. A first-order low-pass filter (along the time axis) is applied to this ratio. The distorted pitch power density in each frame, n, is then multiplied by this ratio, resulting in the partially gain compensated distorted pitch power density $PPY'_{WIRSS}(f)_n$.

After partial compensation for filtering and short-term gain variations, the original and degraded pitch power densities are transformed to a Sone loudness scale using Zwicker's law [13].

$$LX(f)_n = S_l \cdot \left(\frac{P_o(f)}{0.5}\right)^{\gamma} \cdot \left[\left(0.5 + 0.5 \cdot \frac{PPX'_{WIRSS}(f)_n}{P_o(f)}\right)^{\gamma} - 1\right] \tag{3.4}$$

with $P_o(f)$ the absolute threshold, S_l the loudness scaling factor and γ the Zwicker power [12].
The resulting two-dimensional arrays $LX(f)_n$ and $LY(f)_n$ are called *loudness densities*.

The signed difference between the distorted and original loudness density is then computed. When this difference is positive, components such as noise have been added. When this difference is negative, components have been omitted from the original signal. This difference array is called the raw *disturbance density*.
The minimum of the original and degraded loudness density is computed for each time-frequency cell. These minima are multiplied by 0.25. The corresponding two-dimensional array is called the mask array. The following rules are applied in each time-frequency cell:

- If the raw disturbance density is positive and larger than the mask value, the mask value is subtracted from the raw disturbance.

- If the raw disturbance density lies in between plus and minus the magnitude of the mask value, the disturbance density is set to zero.
- If the raw disturbance density is more negative than minus the mask value, the mask value is added to the raw disturbance density.

The net effect is that the raw disturbance densities are pulled towards zero. This represents a dead zone before an actual time frequency cell is perceived as distorted. This models the process of small differences being inaudible in the presence of loud signals (masking) in each time-frequency cell. The result is a disturbance density as a function of time (window number n) and frequency, $D(f)_n$.

An asymmetry effect is caused by the fact that when a codec distorts the input signal it will in general be very difficult to introduce a new time-frequency component that integrates with the input signal, and the resulting output signal will thus be decomposed into two different percepts, the input signal and the distortion, leading to clearly audible distortion. When the codec leaves out a time-frequency component, the resulting output signal cannot be decomposed in the same way and the distortion is less objectionable. This effect is modelled by calculating an asymmetrical disturbance density $DA(f)_n$ per frame by multiplication of the disturbance density $D(f)_n$ with an asymmetry factor. This asymmetry factor equals the ratio of the distorted and original pitch power densities raised to the power of 1.2. If the asymmetry factor is less than 3, it is set to zero. If it exceeds 12, it is clipped at that value. Thus, only those time-frequency cells remain, as non-zero values, for which the degraded pitch power density exceeded the original pitch power density.

The disturbance density $D(f)_n$ and asymmetrical disturbance density $DA(f)_n$ are integrated (summed) along the frequency axis using two different Lp norms:

$$D_n = M_n \sqrt[3]{\sum_{f=1,\ldots NumberBarkbands} (|D(f)_n| \cdot W_f)^3} \tag{3.5}$$

$$DA_n = M_n \sum_{f=1,...NumberBarkbands} (|DA(f)_n| \cdot W_f) \tag{3.6}$$

with M_n a multiplication factor, and W_f a series of constants proportional to the width of the modified Bark bins [12].
After this multiplication the frame disturbance values are limited to a maximum of 45. These aggregated values, D_n and DA_n, are called *frame disturbances*.

Next, the frame disturbance values and the asymmetrical frame disturbance values are aggregated over split second intervals of 20 frames overlapped 50 per cent. A series of operations are computed [12], till two values of the average disturbance and of the average asymmetrical disturbance are found: the numbers α and β.
The final PESQ score is a linear combination of α and β. The range of the PESQ score is -0.5 to 4.5, although for most cases the output range will be a listening quality MOS-like score between 1.0 and 4.5, the normal range of MOS values found in an ACR experiment.

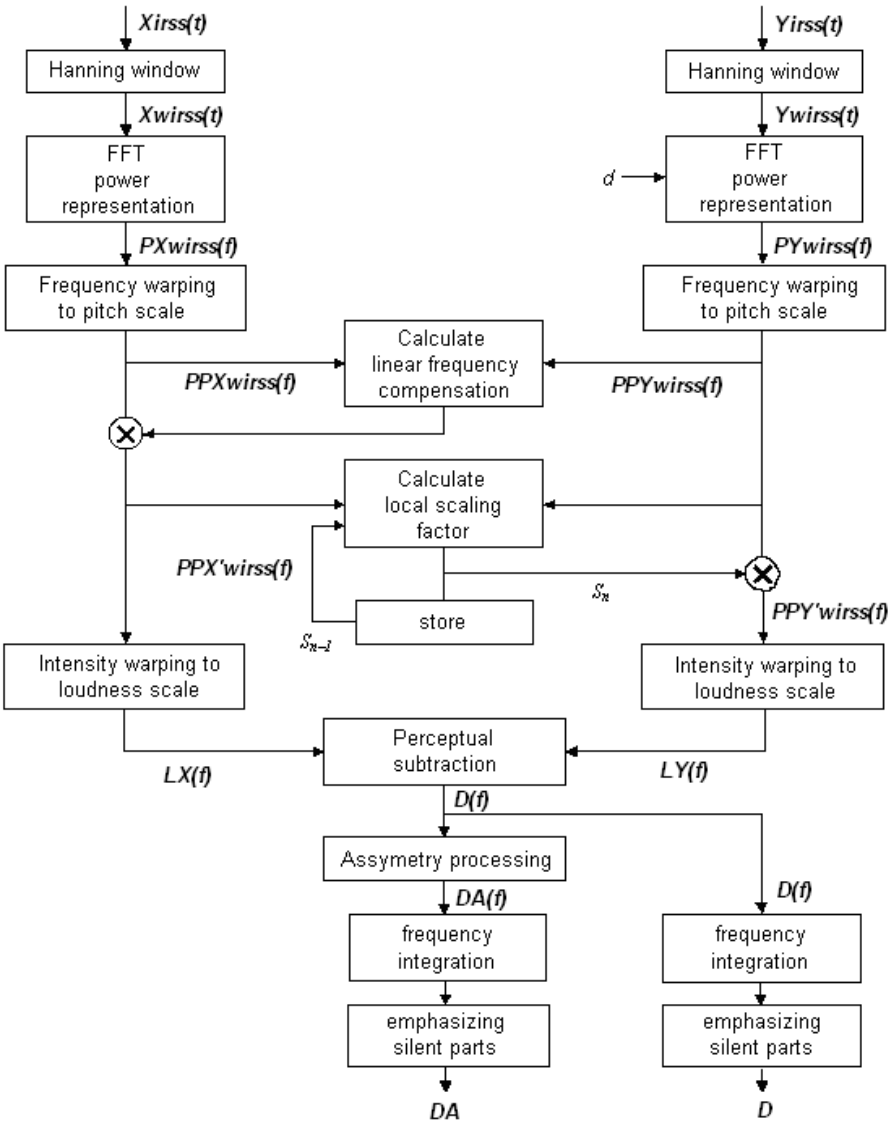

Figure 3.9: Overview of the perceptual model.

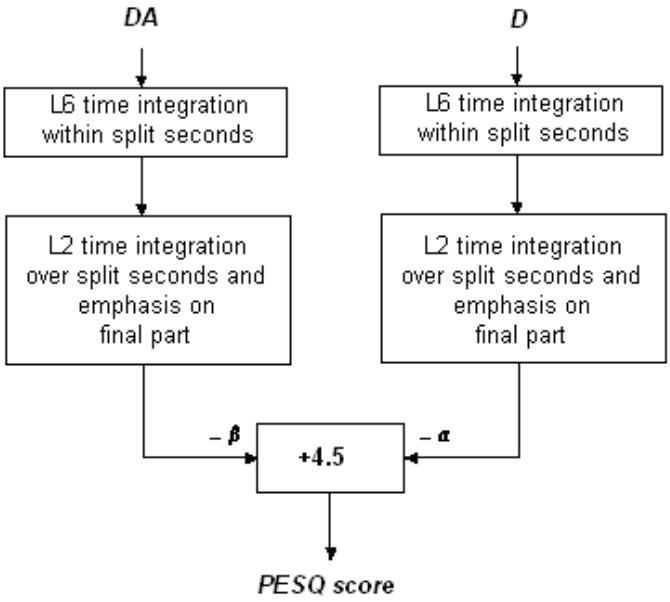

Figure 3.10: Overview of the perceptual model.

Chapter 4

Audiovisual Quality

Introduction

In order to predict the audiovisual quality of a multi-media system it is necessary to build a metric that takes into account both the audio quality and the video one.

When assessing a multi-modal system, one can not model it only as a simple combination of mono-modal models, because the pure combination of audio and video models does not give a robust perceived-quality performance metric [14]. So, it is imperative to take into account the cross-modal interaction between audio and video mode.

Several auditory and visual models are often utilized as basis for multi-modal predictions. They consider how the audio and video signals are perceived by people. In this way the audio and video signals are perceptually weighted before they are combined in a multi-modal model. But the multi-modal model must also account for cross-modal interactions as well as task influences in order to give a task related perceived performance metric. Perceptual interaction is indeed dependent on the nature of the task undertaken.

In multi-media systems, video and audio mode not only interact, but there is even a *synergy* of component media.
It is an human ability to make up for the lack of information from one sense with the other senses. For example, the blind has stronger hearing ability than ordinary

people. A similar observation may be made on a multi perceptual model. In video-telephone system, for instance, even if the quality of the video stream is somewhat low, the voice stream with good quality can compensate for the degradation of the overall perceptual quality, and vice versa [15]. In other words, we guess media compensate for each other from a perceptual point of view.

4.1 Multi-modal modelling

The communications industry is rapidly evolving beyond the traditional audio-only systems that have been with us for over a hundred years. To provide and maintain required levels of customer satisfaction in an efficient manner, it is desirable to have objective measures of quality that relate to perceived performance. For multi-media systems, as in mobile applications, multi-sensory models will be required [17]. Research is being directed towards developing a multi-modal model that can be used to predict audio-video quality.

Understanding and predicting multi-modal performance is particularly difficult when a task situation is dynamically varying - which is not unlikely. Perceptual interaction, and the attention given to particular aspects of performance, is dependent on the nature of the task undertaken. It is anticipated that to understand and eventually model multi-media perception it will be necessary to understand and model the underlying cognitive processes. The use of knowledge about the underlying cognitive mechanisms to inform model structure has two advantages. Firstly, the model is made more representative of actual human quality perception. Secondly, the critical perceptual components of the model can be identified and weighted according to the principles of human information-processing.

The multi-modal model is so obtained combining perceptually weighted signals that come from the auditory and visual models. The perceptual signals are combined in different ways dependent on the task of the audiovisual sequence. The cross-interaction and the mutually compensation between audio and video mode are indeed different for different tasks.

4.2 Mutually compensatory property

In the previous studies of audiovisual quality [15], it was observed a phenomenon of mutually compensation between audio and video quality. Under mutually compensation, one means that it is possible to obtain the same audiovisual quality with different combinations of audio and video quality. The importance of this property is evident. It can be exploited to find out the best way to guarantee a good audiovisual quality with the maximum saving of resources.

Therefore we want to design an audiovisual model that takes into account this synergy of component media. It is usual to indicate the quality of component media in a model with the *mean opinion score (MOS)* of people's evaluation. In our work the *MOS* ranges between 1 and 5, where 5 is the best quality (see chapter 5).

In order to reflect a synergistic effect of the component media on user-level quality of service, we must design an audiovisual model given by the combination of MOS_a and MOS_v indicating respectively the audio and video quality. In particular with the use of a mix-term given by the product $MOS_a \cdot MOS_v$, it is evident that increasing MOS_a and decreasing MOS_v or vice versa, MOS_{av} can remain the same.

We expect also that the values of MOS_a and MOS_v under which a mutually compensatory property can be recognized, depend on the task. In our subjective tests, for instance, we have tested both clips with speech and clips with music. It is clear that the audio has different importance in these cases. In a video clip the sound and therefore the MOS_a could have more importance than in a video call. So even the mutually compensation can be observed at different levels of audio and video quality.

Anticipating part of our tests results, we can show an example of mutually compensatory property. The Figure 4.1 indicates the relation between MOS_a, MOS_v and MOS_{av} for the video call studied in our test (see chapter 5). The two columns of the histogram marked with green color show a mutually compensatory

effect. A MOS_{av} of about 2.8 is obtained with a MOS_a of 4.5 and a MOS_v of 2.5,

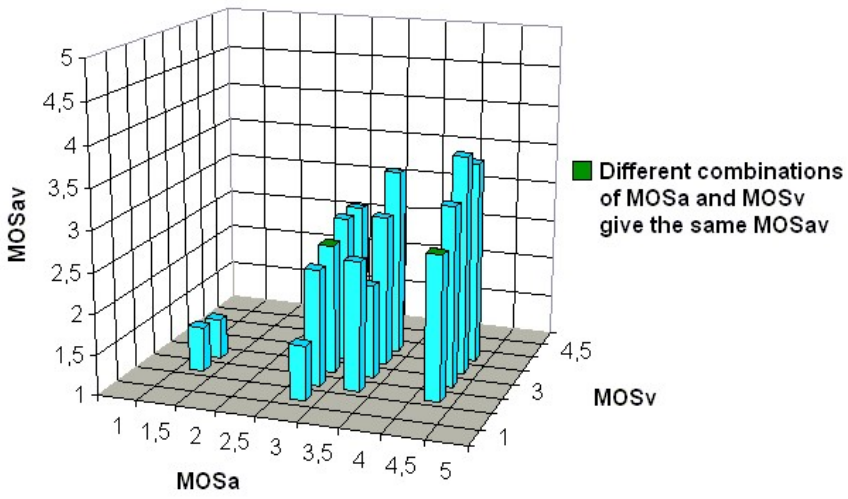

Figure 4.1: Mutually compensatory property for video call.

Chapter 5

Subjective tests

Introduction

In order to study how to predict the subjective perceptual quality evaluation, we have chosen to perform several audiovisual subjective tests. As our study regards the quality in UMTS packet switched streaming service, we have chosen to recreate the real conditions of the UMTS services in our tests. This is the principle that we have adopted in the tests setup. This is the reason why we have used the Absolute Category Rating method and why all the sequences were displayed on a UMTS mobile phone. Even the utilized resolution and the bit rate combinations reflect real UMTS streaming configurations.
The source material for the tests is composed of three multimedia clips representative of typical UMTS multimedia scenarios: a cinema trailer, a video clip and a video call.
All the tests were performed according to ITU-T Recommendation [18].

In order to obtain the subjective ratings for audio quality, we have performed even audio-only tests on the three sequences, adopting the same test methodology and recommendation of the audiovisual tests.

5.1 Audiovisual tests

5.1.1 3G streaming scenario

Three typical UMTS streaming scenarios are cinema trailer, video clip and video call. The three sequences chosen in our test have very different characteristics (Figure 5.1).

Figure 5.1: Test clips: cinema trailer (a), video clip (b), video call (c).

In the *cinema trailer* scenes change very rapidly and are shot from several angles; the music is only an accompaniment and no voice is present. In the *video clip* is instead the music, instrumental and with voice, in foreground. In the scenes there are rapid movements and rapid zoom in/ zoom out of the camera. The *video call* is the simplest scenario. The camera is fix and only slow and small movements are present in the scene. In this case the audio material is a speech monologue, without music.

All the multimedia clips are nearly 8 sec long [18]. The clip length is adjusted to the scene, in order not to break the scene integrity. The source material was originally in an high quality format. Using QuickTime Pro version 6.5 we could encode our clips with several codecs and combinations of bit rate. We have chosen to utilize H.263 and MPEG-4 as video codecs, and AMR and AAC as audio codecs. All these codecs are supported by 3GPP file format. Even the chosen bit rate combinations are configurations that really exist in UMTS packet switched streaming services. The utilized combinations (Table 5.1) were chosen because

they are significative to study the way with which the audio quality and the video quality influence the total audiovisual quality, and to highlight a possible mutually compensation effect between the two single media.
Through QuickTime Pro version 6.5, the frame resolution was made QCIF, size supported by UMTS mobile phones, and the frame rate was set to 8 fps, the most used rate for UMTS video services [19].

Table 5.1: Codecs and bit rate combinations used for subjective tests; 36 combinations for "cinema trailer" and "video clip", 30 combinations for "video call"; 102 total encoded sequences.

		MPEG-4 / H.263 total bit rate		
		56 kbps	75 kbps	105 kbps
AAC	8 kbps	clips 1,2,3		
audio bit rate	16 kbps	clips 1,2,3	clips 1,2,3	
	24 kbps	clips 1,2	clips 1,2,3	clips 1,2,3
	32 kbps		clips 1,2	clips 1,2,3
	48 kbps			clips 1,2
AMR	5,9 kbps	clips 1,2,3	clips 1,2,3	clips 1,2,3
audio bit rate	7,9 kbps	clips 1,2,3	clips 1,2,3	clips 1,2,3
	10,2 kbps	clips 1,2,3		
	12,2 kbps		clips 1,2,3	clips 1,2,3

5.1.2 Test setup

The subjective audiovisual tests were performed according to ITU-T Recommendation [18], using Absolute Category Rating method (ACR). According to this method, only the codified sequences are displayed. Indeed in the real UMTS services, resolution and bit rate are limited; so only codified and degraded sequences are available to the customs. It is therefore more reasonable to adopt

ACR method than DCR (Degradation Category Rating) method, where also the original sequence without degradation is displayed.

After each presentation the subjects are asked to evaluate the quality of the sequence presented. For this we have utilized a 5 grade MOS scale (1-bad, 2-poor, 3-fair, 4-good, 5-excellent). A scale with higher discriminative power would have caused a too high variance in the results. Indeed subjects had not the original sequence as reference (since ACR method was adopted); so their evaluation suffers already from a rather high variance.

In order to study how the audiovisual quality is perceived in the UMTS services, we have chosen to perform the test on a UMTS Sony Ericsson Z1010 phone. The viewing distance from the phone was not fixed but we have noticed that subjects were comfortable to take the phone at a distance of 20-30 cm. Moreover, since one of our intention is to study the relation between audio quality and audiovisual quality, we have chosen to take all the tests with Sony Ericsson high quality stereo handset in a quiet environment. In this way we could obtain a higher correlation between the objective audio quality measured with our parameters and the perceived audio quality. As a consequence, it was possible to obtain a higher correlation between the perceived audio quality (MOS_a) and the subjective audiovisual quality (MOS_{av}).

Before starting the experiment, a training session of three sequences with different characteristics was taken where test persons could adjust the volume as they preferred. Then the 102 sequences of the test were presented in an arbitrary order, with additional condition that the same sequence (even differently degraded) did not appear in succession. It is indeed simpler to realize a difference in the quality between two sequences with same content but differently codified, if they are consecutively displayed. Therefore it is clear that the consecutive repetition of a sequence with different degradation influences the evaluation of the quality. In order to increase the reliability per subject, two rounds of each test were taken. The whole test lasts about 40 minutes.

To evaluate the subjective perceptual audiovisual quality, we have chosen a

varied group of people, so that they can be representative of the real customs of the UMTS services. We worked with 20 unpaid test persons. The chosen group ranged different ages (between 17 and 30), sex, education and experience in imaging processing. Before the tests, subjects were asked to fill out a small questionnaire to obtain this information. A set of instructions was told to the subjects, where we have demanded the use of the whole scale and that each clip is viewed only once before the evaluation.
To assure a general value to our results, we have performed the test twice per person (i.e. 40 tests in total) and in the further processing of data results we have rejected the sequences which were evaluated with individual variance higher than one (7% of the whole results).

5.1.3 Tests results

Observing the tests results we have noticed a great difference between video call on one hand, and video clip and cinema trailer on the other.

The mean of the MOS_{av} for *video call* is 2.75. In this case one obtain good results using AMR code (mean value equal to 2.753). AMR was indeed designed as speech codec and in the video call only speech is present (no music). Utilizing 5.9 Kbps or 7.9 Kbps for audio rate, the audiovisual quality evaluation is practically the same. A small difference in the quality evaluation one observe in the cases of 7.9 or 12.2 Kbps as audio rate. This fact is shown in Figure 5.2. Here we have grouped the video rates of 99.1, 97.1 and 92.8 Kbps as 90 Kbps, 69.1, 67.1 and 62.8 Kbps as 60 Kbps, 50.1, 48.1 and 45.8 Kbps as 50 Kbps. On the axis of the audio bit rate, we have grouped 10.2 and 12.2 Kbps. One can notice that under the same total bit rate, MOS_{av} is maximum where the audio bit rate is maximum. Therefore the audio quality has in the video call a great importance.

Video clip and *cinema trailer* have a similar behaviour because both have a background of music. The mean of the MOS_{av} for video clip and cinema trailer is 3. In this case the use of AMR codec induces very bad results. Therefore it makes

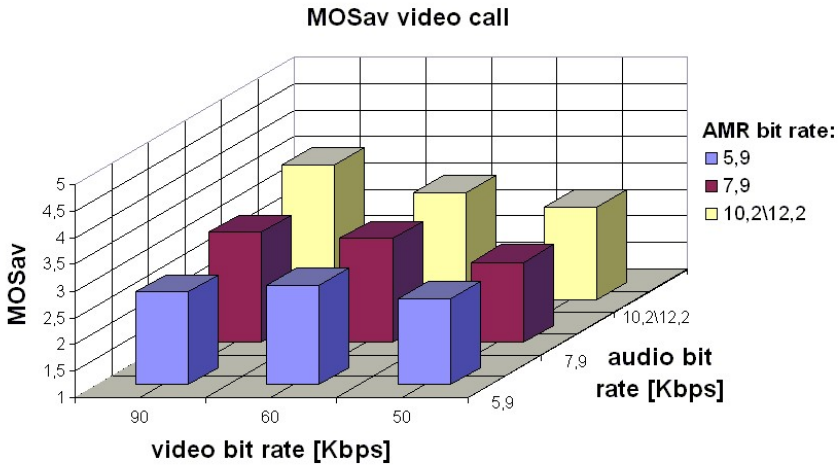

Figure 5.2: MOSav for video call as function of video bit rate and audio bit rate, using only AMR audio codec.

no sense utilizing the AMR speech codec for music; the AAC codec is instead very useful. Moreover one can notice that even with AAC, the use of a too low audio bit rate causes always bad results. The use of 8 or 16 Kbps for audio is unacceptable. But when the audio bit rate is high (from 24 up to 48 Kbps), the importance of the video bit rate increases. If one utilizes 105 Kbps in total, one obtains the best result when video bit rate is maximum, i.e. with 24 Kbps for audio and 81 Kbps for video (see Figure 7.8). Instead when the total bit rate is 75 Kbps, one obtains the best result in a trade off between bit audio and bit video, i.e. with 24 Kbps for audio and 51 Kbps for video.

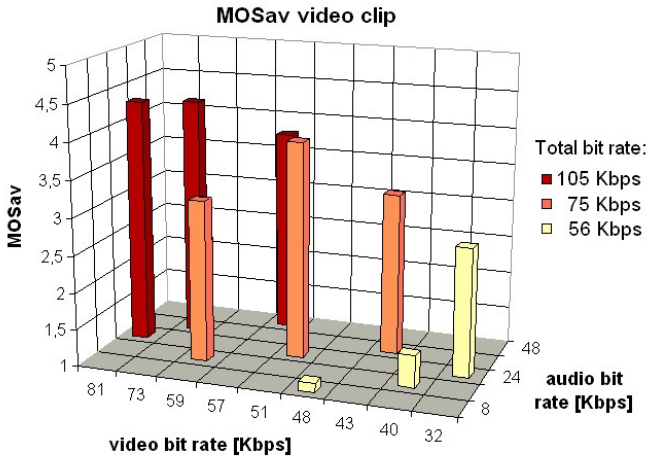

Figure 5.3: MOSav as function of video/audio bit rate, using only AAC codec.

5.2 Audio tests

The audio-only tests were conducted as the audiovisual tests. During the training session of three sequences the subjects were allowed to adjust the volume of the headset to a comfortable level. The audio clips were the same used in the audiovisual tests, too, but the total number of the tested sequences is very smaller: they are 26 audio clips, as shown in Table 5.2.

The adopted methodology was Absolute Category Rating, that suits for the real conditions of a UMTS service, as explained for the audiovisual tests. The order of the tests sequences was random, but the same sequence (even differently degraded) appeared never in succession. The evaluation scale was a 5 grade MOS scale (1-bad, 2-poor, 3-fair, 4-good, 5-excellent).

Also in this case, one observes a great similarity between video clip and cinema trailer results, and a great difference with video call results. This behaviour is certainly due to the fact that the audio material is speech in the video call, whereas it is music in video clip and cinema trailer.

Observing the audio results, it is evident that AMR codec operates very efficiently in the *video call* (Figure 5.4). If 5.9 Kbps are utilized as audio bit rate, the perceived audio quality is very good: MOS_a is 3.5. The differences in MOS_a

Table 5.2: Codecs and bit rates used for audio-only tests. Clips 1, 2, 3 are respectively cinema trailer, video clip and video call.

AAC	8 kbps	clips 1,2,3
audio bit rate	16 kbps	clips 1,2,3
	24 kbps	clips 1,2,3
	32 kbps	clips 1,2,3
	48 kbps	clips 1,2
AMR	5,9 kbps	clips 1,2,3
audio bit rate	7,9 kbps	clips 1,2,3
	10,2 kbps	clips 1,2,3
	12,2 kbps	clips 1,2,3

utilizing 5.9, 7.9, 10.2 and 12.2 Kbps are not considerable. The mean value of MOS_a is 3.5 using AMR codec.
AAC codec gives instead different results in dependence on the utilized bit rate: MOS_a is smaller than 2.5 if AAC operates at 8 or 16 Kbps, whereas it is higher than 4 if AAC utilizes 24 or 32 Kbps.

In the *video clip* and *cinema trailer* very bad results are obtained using AMR codec: MOS_a is always smaller than 2.2. The mean value in this case is only 1.67. High values of MOS_a are instead gotten with the AAC codec. The best evaluations reach even 4.35. The mean value of MOS_a is 3.16 using AAC (Figure 5.5).

All the audio tests results are attached in appendix.

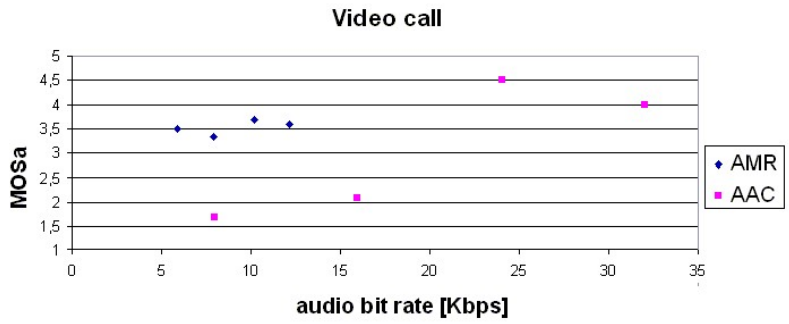

Figure 5.4: MOSa function of audio bit rate for video call.

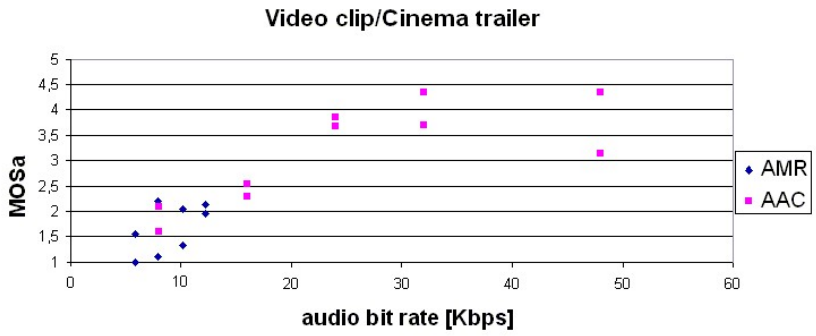

Figure 5.5: MOSa function of audio bit rate for video clip/cinema trailer.

Chapter 6

Metrics design

Introduction

In this chapter the video, audio, and audio-video metrics are defined. They allow to predict the video, audio and audiovisual quality through the use of the objective parameters, partly explained in the previous chapters (chapters 2 and 3). We have implemented all the parameters utilized in our metrics. The whole matlab implementation is enclosed in appendix.

At first, according to the ANSI standard [16], we have developed our visual metric, which enables us to predict the MOS_v.
The second step was creating audio metrics different for speech and music, so that the correlation with the results of the performed audio tests was maximum.
At last, utilizing the so predicted MOS_v and MOS_a, we have developed several audiovisual metrics in order to propose the one that allows to predict the subjective quality evaluation at best.
Through these steps we have reached the goal of our work: predicting the subjective audiovisual quality of a clip from its objective characteristics. The limitation of resources in a UMTS service causes indeed a degradation of the clip. From this degradation one can predict the subjective audiovisual quality evaluation.

6.1 Video metric

The video metric here defined concerns the parameters ***sigain***, ***siloss***, ***hvgain*** and ***hvloss*** explained in chapter 2. They are connected with gain and loss of intensity and orientation of the spatial activity. They consider if the codec operates through an edge sharpening and if a phenomenon of blockiness or blurring happens.

The idea of this metric comes from the ANSI standard [16]. Here from seven objective parameters one obtains a *video quality parameter* VQ between 0 and 1, where 0 indicates that the processed video is not degraded (best quality) and 1 indicates the worst quality. We have simplified this video model, utilizing only the four most important parameters. So our *video quality parameter* becomes:

$$VQ = -0.2097 \cdot siloss + 0.5969 \cdot hvloss + 0.2483 \cdot hvgain - 2.3416 \cdot sigain|^{0.14} \quad (6.1)$$

where the parameter *sigain* is clipped at an upper threshold of 0.14 which indicates the maximum improvement of the video quality observed in the encoded sequences. The *sigain* parameter is the only quality improvement parameter in the model (since the sigain parameter is positive, a negative weight results in negative contributions to VQ which produce quality improvements).

To obtain MOS_v, it is necessary to map VQ into the range between 1 and 5, where 1 indicates the worst quality and 5 the best one.

$$MOS_v = (1 - VQ) \cdot 4 + 1 \quad (6.2)$$

In order to verify the effectiveness of our metric, we have compared our predicted MOS_v with the results of subjective video tests. For this goal we have not performed video tests on purpose, but we have utilized the results of video tests already performed on different video sequences. They are four different sequences with very different characteristics; a panorama and a soccer sequence with a lot of movements of the camera, two sequences with a woman and a man who speak, with static camera [20]. The correlation between our predicted MOS_v and the evaluation given in the tests is 91%.

6.2 Audio metrics

In our tests we have noticed that the audio evaluation is very different in the case of speech and music. Music has indeed a higher complexity than speech. This fact leads to a difference in the quality evaluation (see chapter 5). For this reason we have chosen to develop different objective parameters and metrics for speech and music quality evaluation. Designing two different metrics for speech and for music, one obtains the best correlation with the audio subjective evaluation.

6.2.1 Speech quality evaluation

Our speech metric is based on the use of the ***AD*** parameter normalized between 0 and 1. It measures the dissimilarities between the original and the compressed speech signals.

Designing the audio quality metric we have noticed a difference in the subjective audio evaluation when the sequences are encoded with the codec AMR or AAC. The maximal correlation between our quality prediction and the measured MOS_a (mean opinion score for audio) is obtained with a translation of the audio metric in the two cases of AMR (equation 6.3) and AAC coding (equation 6.4).

$$MOS_a = -6.996AD^2 + 10.95AD + 1.165 \tag{6.3}$$

$$MOS'_a = -6.996AD^2 + 10.95AD + 0.370 \tag{6.4}$$

The reason for this translation is in the way with which the codecs operate. AAC codec utilizes a wider range of frequency; so it degrades objectively the signal less than AMR. But the audio subjective evaluation is however higher for AMR. Indeed AMR is a codec designed for speech. It degrades the signal in a way that human's ears do not perceive. Therefore, although the objective degradation is stronger for AMR, the subjective speech evaluation is higher.

With this division, one obtains a 98% correlation for AMR and a 84% correlation for AAC with the tests results.

6.2.2 Music quality evaluation

In order to create a metric to evaluate the music quality, we have developed objective parameters whose ideas come from PESQ method (chapter 3), but with important changes.

Objective parameters

The first part of pre-processing of the original and degraded signal defined in PESQ method is not considered here. PESQ method was indeed defined for speech quality; therefore the filers used in PESQ can not suit for music signals that spread over more frequencies than speech.

In order to find the delay between the original and the degraded audio signal, we have developed two Matlab functions. They are divided into crude delay algorithm and fine delay algorithm. The implementation follows the instructions in chapter 3.

We have also maintained the same perceptual model as in PESQ method. The original and degraded signals are broken into frames of 32 ms, 50% overlapped, multiplied with Hann windows. Then, through the sum of the squared real and squared imaginary parts of the complex FFT components, we have obtained the power representations of the two signals (see chapter 3). In this power representations we have both temporal and frequency information of the original and encoded signals, because we have a division in temporal frames and a division in frequency through the Fast Fourier Transform.
The frequency scale was afterwards transformed in Bark scale that is more suitable for the description of the human hearing system; the power representations of the signals are so transformed into *pitch power densities*.

Meaning over time the pitch power densities, we have obtained the *mean power* of the original and degraded signals as function of the Bark frequency. In Figure 6.1 it is shown an example of mean power over frequency for an original and a degraded audio signal.

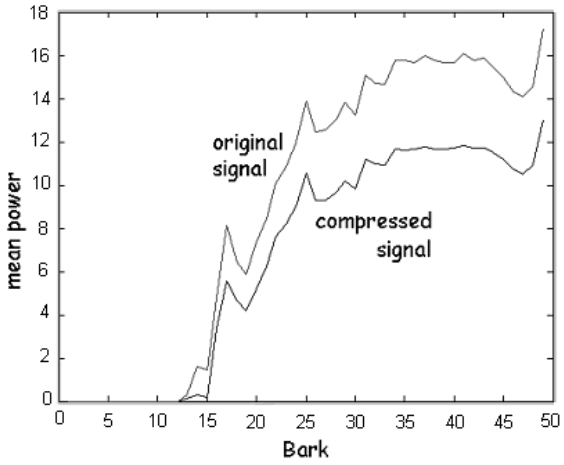

Figure 6.1: Mean power of an original and a degraded audio signal.

Our first parameter, called ***Integrated Frequency Distance (IFD)***, is obtained by the integration over frequency of the difference between the original mean power and the degraded one. The greater the difference between the original and degraded signals, the higher the parameter IFD.

For the definitions of the other two parameters, we have computed other transformations on the pitch power densities according to PESQ method. In particular these transformations take into account if silent frames are present in the audio signals, and perform a coefficient in order to compensate the disturb to the listener created by severe filtering. For the detailed explanation of these steps one can see chapter 3 and matlab implementation in appendix.

After this signals processing, the original and degraded pitch power densities are transformed to a Sone loudness scale (scale for perceptual loudness) using Zwicker's law (**??**, chapter 3), so that the two *loudness densities* are obtained.

The difference between the distorted and original loudness density is then computed. From this difference, two disturb phenomenons can be recognized: the noise and the loss of time-frequency components caused by the encoding. The *disturbance density* and the *asymmetrical disturbance density* parameters account these disturbs. The disturbance density is given by the difference between the two loudness densities, after they have been processed with some mathematical oper-

ations (chapter 3). The asymmetrical disturbance density is obtained multiplying the disturbance density with an asymmetrical factor (chapter 3).
The disturbance density and the asymmetrical disturbance density are then integrated along the frequency axis. At last, through a division in syllables of 20 frames, a series of mathematical operations (see matlab implementation) and an integration over the time, our two parameters, ***Disturbance Indicator*** D_{ind} and ***Asymmetrical Disturbance Indicator*** A_{ind} are obtained.

Non-speech metric

Utilizing the three audio parameters described above, we have designed a non-speech metric used to predict the quality of our tested video clip and cinema trailer.
The IFD parameter indicates how much the power descriptions of the original and degraded signals diverge. Therefore it is in inverse relation to MOS_a (equation 6.5): if the power descriptions of the two signals diverge very much, the encoded signal is very degraded, i.e. the audio quality of the processed signal is low. The other two parameters, denoted as *D_ind* and *A_ind* (disturbance indicators), consider instead how much the presence of noise and the loss of time-frequency components influence the audio quality. They are proportional to MOS_a through coefficients that are negative; so the greater the disturbance parameters, the lower the audio quality.

$$MOS''_a = 3.1717 + \frac{4.8809}{IFD} - 0.3562 \cdot A\_ind - 0.0786 \cdot D\_ind \tag{6.5}$$

In this way one obtains a 91% correlation with the subjective evaluation. The scatter plot of the audio model output versus the subjective MOS_a is shown in Figure 6.2.

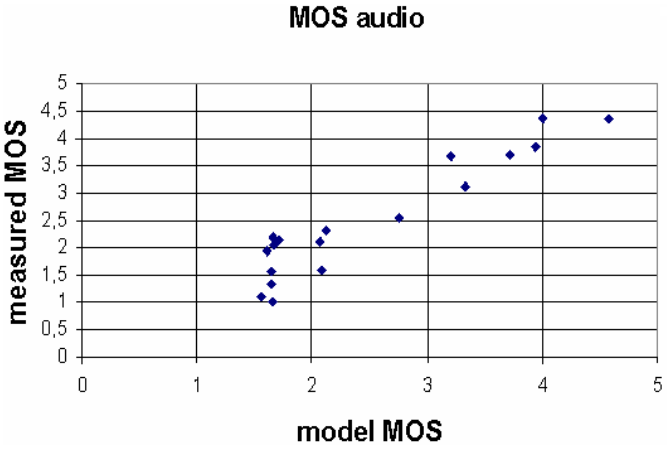

Figure 6.2: Audio model output versus the subjective MOS_a.

6.3 Audiovisual model

6.3.1 Dependence of the model on the quantity of information

In order to design an audiovisual metric to predict the quality of our clips, the first step was to observe the results of our audiovisual subjective tests. We have noted that the test persons' rating is very different for "film trailer" or "video clip" on the one hand, and "video call" on the other hand. This fact is caused by a loss of critical ability of the subjective judgment when the media has poor information contents. The video call is characterized by slow and small movements, fix camera and a uniform tone of voice. The tests results for video call show that the evaluation range in this case is small; the 66,7% of the clips are evaluated with a MOS_{av} between 2 and 3.

If we encode a "video call" with high bit rate (105 Kbps), i.e. high resolution, the objective quality is certainly very high. If we encode a "video clip", characterized by fast movements and many scene changes, with the same bit rate, the objective parameters do not indicate a quality so high as in the video call, but the subjective evaluation is higher in this second case. People rate the video clip quality higher because their judgment is based on more information. The high entropy causes a

very high evaluation when the total bit rate used is high (i.e. in the best cases) and a very low evaluation when the total bit rate used is low (for example 56 Kbps). On the contrary, the low entropy (case of video call) causes a reduction in the evaluation range. If we consider the MOS range between 2 and 3, we find the 66,7% of the sequences for video call, but only the 44,4% for the video clip and cinema trailer.

This diversity in the subjective evaluation between video clip/cinema trailer and video call has induced us to create for speech and music different metrics with the same form but different coefficients' values.

6.3.2 Audiovisual metrics

The first model that we have designed is a plane in the space given by MOS_v, MOS_a and MOS_{av} (equation 6.6). The slopes relative to MOS_v and MOS_a are different. With this model one obtains already a good approximation in both cases of music and speech. But the correlation between MOS_{av} of the model and the subjective MOS_{av} is higher in the other models.

$$MOS_{av}^{I} = K + A \cdot MOS_a + V \cdot MOS_v \tag{6.6}$$

$$MOS_{av}^{II} = K + AV \cdot MOS_a \cdot MOS_v \tag{6.7}$$

$$MOS_{av}^{III} = K + A \cdot MOS_a + V \cdot MOS_v + AV \cdot MOS_a \cdot MOS_v \tag{6.8}$$

$$MOS_{av}^{IV} = K + A \cdot MOS_a + V \cdot MOS_v + AV \cdot MOS_a \cdot MOS_v + A' \cdot MOS_a^2 + V' \cdot MOS_v^2 \tag{6.9}$$

In the other models (equations 6.7, 6.8, 6.9) we have introduced a mix term of second degree. It allows the increasing of the correlation and reflects the synergistic effect present between video and audio quality. The synergy of component media is the most important feature of multimedia: when the video quality is bad, but

Table 6.1: Coefficients and correlation of "video call" model.

equation number	K	A	V	AV	A'	V'	correlation
(6.6)	-0,4934	0,5420	0,4327	/	/	/	0,8800
(6.7)	0,9987	/	/	0,1536	/	/	0,8915
(6.8)	0,6313	0,2144	0,0124	0,1184	/	/	0,9023
(6.9)	0,5723	9,6508	0,2686	0,2244	-0,0171	-0,0940	0,9057

Table 6.2: Coefficients and correlation of "cinema trailer"/"video clip" model.

equation number	K	A	V	AV	A'	V'	correlation
(6.6)	-1,5025	0,7380	0,7411	/	/	/	0,8879
(6.7)	0,9135	/	/	0,2329	/	/	0,8415
(6.8)	-0,9222	0,5691	0,5064	0,1697	/	/	0,9106
(6.9)	-1,1895	0,5947	0,7126	0,0677	-0,0031	-0,0395	0,9117

the audio quality is good, this can compensate the degradation of the overall perceptual quality, and vice versa (chapter 4). This property is good described by a parabolic form, given by the last three equations.

All these models are designed both for speech (video call) and for music (video clip/cinema trailer). The coefficients' values are shown in Table 6.1 and Table 6.2.

The highest correlations in both cases are obtained by models 8 and 9. In Figure 6.3 the correlation in each case is visualized.

The correlations in the models 8 and 9 are almost equal, but the model 9 with the two terms of second degree for MOS_a and MOS_v, is much more complex than model 8. Therefore the best way to predict the audiovisual quality is the model 8. It is the best trade-off between quality and complexity.

The correlation factor which we have utilized to evaluate how much our metrics

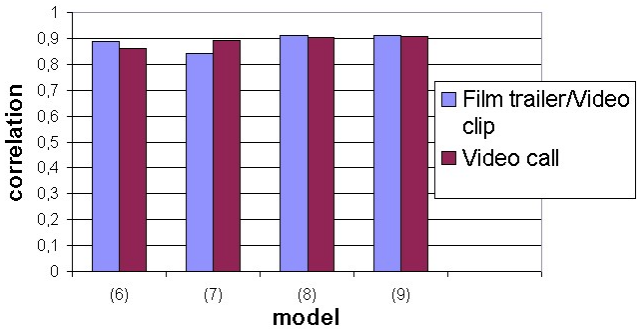

Figure 6.3: Correlations of models (MOS_{av}^{I}, MOS_{av}^{II}, MOS_{av}^{III}, MOS_{av}^{IV}) with subjective tests.

fit the subjective tests results, is defined as follows:

$$r = \frac{\mathbf{x}^T\mathbf{y}}{\sqrt{(\mathbf{x}^T\mathbf{x})(\mathbf{y}^T\mathbf{y})}}. \tag{6.10}$$

In our case vector $\mathbf{x}$ corresponds to MOS_{av} and vector $\mathbf{y}$ corresponds to the metric prediction.

Chapter 7

Tests results and metric evaluation

Introduction

In the previous chapter we have designed some audiovisual metrics in order to predict the audiovisual quality. We have maintained different metrics for video call and video clip/cinema trailer because they represent two complete different scenarios.

In this chapter we want to examine the tests results in order to find which bit rate combinations and which codecs bear the best audiovisual quality.
The utilized total bit rates are 56, 75 and 105 Kbps. A part of this bit rate is reserved to audio, and the other part to video. The two audio codecs are AMR and AAC; they were used at 5.9, 7.9, 10.2 and 12.2 Kbps for AMR, and 8, 16, 24, 32, 48 Kbps for AAC. The video codecs are H.263 and Mpeg4.

Under these codecs and bit rates, our goal is to find the most efficient way to use the transport channel, which means the combination of audio and video bit rates that causes the highest audiovisual quality.

7.1 Evaluation of speech results

First of all we must remark the reduction of the range used by people in the quality evaluation. This fact is due to the low entropy present in the video call sequence, as already explained in chapter 6.

The influence of the audio quality on the total audiovisual quality is very high in the video call. The principal information is indeed carried by the audio media (the images are static). Moreover the video quality is always fairly good, because the speaker is still and there is no scene changes. So even a low video bit rate is enough to obtain a fairly good quality. Therefore it is clear that the influence of audio quality on the audiovisual quality is greater than that of the video one.

The maximum of MOS_{av} to a fixed total bit rate is where the MOS_a is maxi-

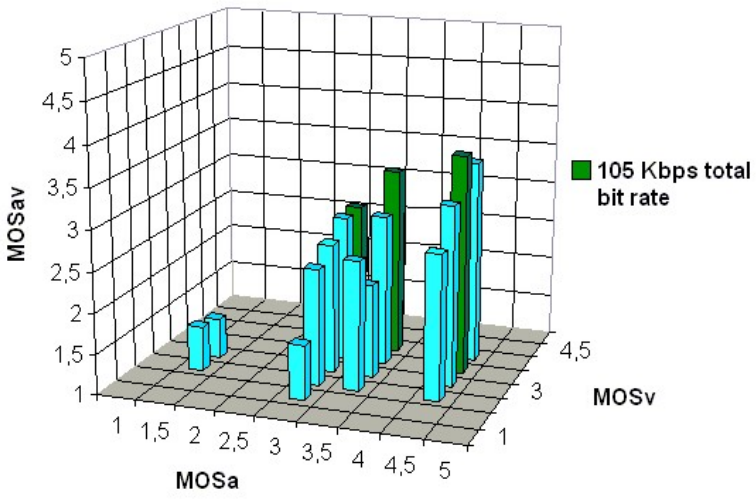

Figure 7.1: MOSav as function of MOSa and MOSv in the video call.

In Figure 7.1 we can also notice the mutually compensation present between audio and video quality : different combinations of MOS_a and MOS_v can give the same MOS_{av}. This effect and the general trend of our data are shown by our models. In particular the Figure 7.2 plots the curve of our best model (best

trade-off between quality and complexity):

$$MOS_{av}^{III} = K + A \cdot MOS_a + V \cdot MOS_v + AV \cdot MOS_a \cdot MOS_v \qquad (7.1)$$

It is a bend plane inclined respect to MOS_a and MOS_v. It is also visualized how much our model data diverge from tests data.

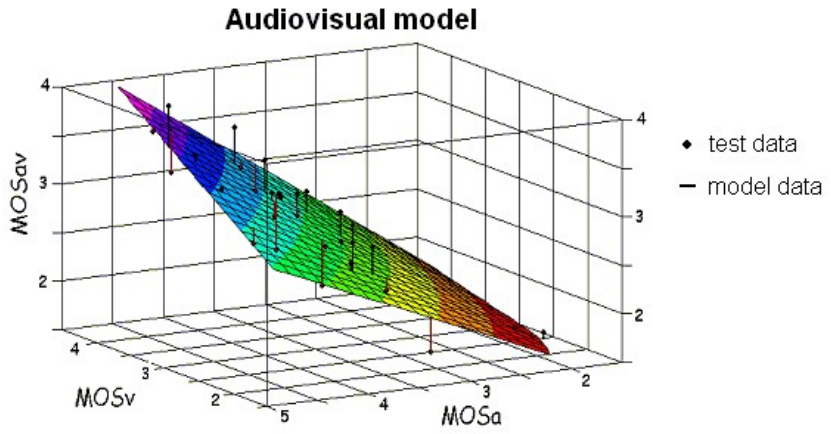

Figure 7.2: Video call model.

The utilized video codecs are Mpeg4 and H.263. Even the use of one of these two codecs does not influence very much the video quality: Mpeg4 and H.263 bring almost the same MOS_v under the same bit rate, as shown in Figure 7.3. This confirms the fact that the video quality is always fairly good because the information contained in the video stream is poor.

We have tested both the use of AMR and AAC as audio codecs. We have encoded 18 video call sequences with AMR and 12 with AAC. In our tests the AMR codec operates at 5.9, 7.9, 10.2 and 12.2 Kbps, while the AAC codec operates at 8, 16, 24 and 32 Kbps. Therefore AAC uses generally a bit rate higher than AMR. Despite this, the mean value of MOS_{av} obtained using AMR is slightly higher than that obtained using AAC. They are respectively 2.79 and 2.72.

The Figure 7.4 shows an important fact. In order to obtain a MOS_{av} of circa 3, 7.9 Kbps are enough using AMR codec. If we want to get the same result with AAC codec, we must use 24 Kbps. Therefore through the AMR codec we can save

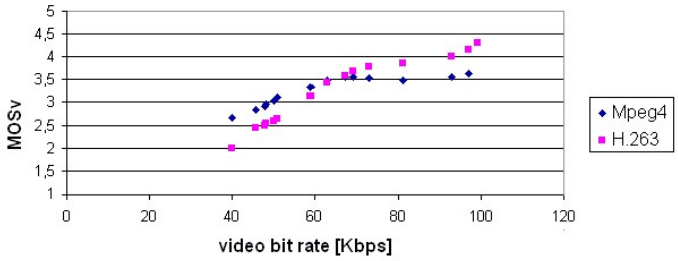

Figure 7.3: MOSv obtained with Mpeg4 and H.263 for video call.

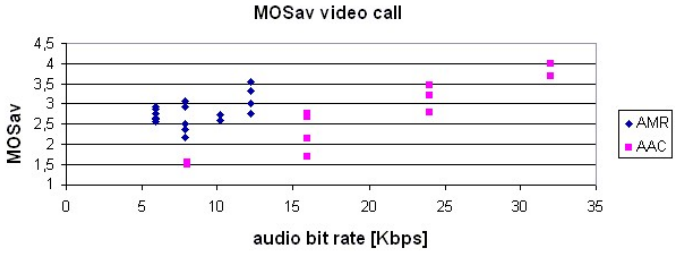

Figure 7.4: MOSav obtained through the use of AMR and AAC codec.

16.1 Kbps.

The Figure 7.4 shows also that both the lower and the upper bounds in the MOS_{av} one gets with AAC codec. The lower bound is given by the use of AAC at 8 Kbps. Certainly it makes no sense to use AAC at this bit rate because one can obtain very better results with 5.9 AMR Kbps.

On the other hand, if we want to obtain the best audiovisual quality we must utilize AAC as audio codec. But in this way we need an high bit rate (32 Kbps). It is instead more convenient to use AMR as audio codec in the video call: in this way we can guarantee a fairly good audiovisual quality already with 5.9 Kbps and so we can save a lot of bits. In other words, the best way to utilize the resources

guaranteing a good quality for video call is the use of AMR as audio codec. In our case it is convenient to use 12.2 Kbps because it produces a quality significantly higher than the other bit rates. Instead using 5.9, 7.9 or 10.2 Kbps, the difference in the quality is not so great. People perceive the quality in these cases almost in the same way. This can be the reason why the use of 10.2 Kbps implies a MOS_{av} smaller than in the other cases.

7.2 Evaluation of music results

Video clip and cinema trailer sequences are scenarios more complex than video call. The video media are rich in movements and scene changes, and audio media are music. This induces people to evaluate the quality using the whole MOS scale (from 1 up to 5), even if the most of results are around 3.

The main difference between video clip and cinema trailer is the importance given to the music. In cinema trailer it is only an instrumental accompaniment to the scenes. In video clip the music is instead in the foreground of people interest. Therefore we expect that the influence of the audio quality on the audiovisual quality is greater in this second case.

In both video clip and cinema trailer we have noticed that the use of H.263 or Mpeg4 as video codec causes a difference in MOS_v and MOS_{av}. This difference is very small for cinema trailer. In video clip, instead, where the scenes are dark and change very fast, Mpeg4 gives very better results both on MOS_v and on MOS_{av}, as shown in Figure 7.5.

Analysing the influence of the audio codec on the audiovisual quality, we notice that AMR codec gives very bad results for music. In this case the MOS_{av} is always below 2.8. On the contrary, when we use AAC codec the subjective evaluation reaches even 4.33. If we consider together video clip and cinema trailer the mean values of MOS_{av} for AMR and AAC are respectively 1.96 and 2.87. Therefore it is evident that although the use of AMR codec allows to save a lot of bits, it does not give sufficient results for the music. AAC guarantees instead very better

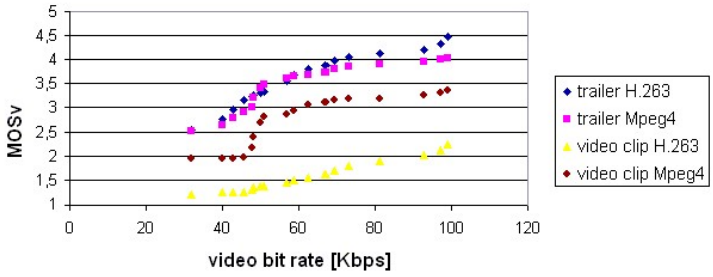

Figure 7.5: MOSv obtained with H.263 and Mpeg4 for cinema trailer and video clip.

results. This fact is highlighted in Figure 7.6 and Figure 7.7, where MOS_{av} is plotted according to the audio bit rate.

Certainly if one wants to use AMR codec, one gets the highest evaluation with 12.2 Kbps. But using only more 3.6 Kbps, that is 16 Kbps with AAC, one gets very better results, especially for cinema trailer.

It is interesting to note that in video music one gets an highest MOS_{av} using 7.9 Kbps with AMR, than using 8 Kbps with AAC. It seems that at very low audio bit rate it is better to use AMR. That is true only for video music, not for cinema trailer. The explanation is in the fact that in video music the audio consists in instrumental music and voice, while in cinema trailer no voice is present. Therefore AMR, that is a speech codec, allows to hear quite good the voice, but not the music. So when AAC codec is not utilized at the best, the quality evaluation for video music with AMR is better than with AAC. Instead for cinema trailer AMR gives never better results than AAC.

However the influence of audio media (as of video media) on the subjective audiovisual quality depends on the total bit rate utilized. The results obtained using AAC for video clip and cinema trailer are visualized in Figure 7.8 and Figure 7.9.

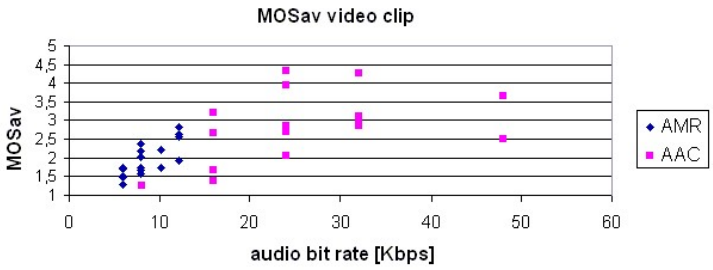

Figure 7.6: MOSav obtained through the use of AMR and AAC codec for video music.

For video clip, under 56 Kbps of total bit rate, one gets the best MOS_{av} using 24 Kbps for audio, that is the maximum bit rate possible for audio in this case. Therefore, if the total bit rate is low, it is more important to allocate the best bit rate for audio. Audio is indeed in the foreground of people interest for video clip, and the video is very complex here. So a bit rate increase may result only in a negligible improvement in video quality for such a scene, while an increase by the same amount can significantly improve the audio.
Under 75 Kbps total bit rate, the best MOS_{av} is instead obtained in a trade off between audio and video bit rate. If the total bit rate increases more, the video media gains importance. Under 105 Kbps, the best evaluation is indeed where the video bit rate is maximum. The best MOS_{av} is obtained with 81 Kbps video bit rate, and 24 Kbps audio bit rate. Here 81 Kbps are enough to improve the video quality, and the audio quality is already fair with 24 Kbps.
In video clip an audio bit rate lower than 24 Kbps is never acceptable. It gives always very bad results, independently from the total bit rate. This fact underlines again the importance of audio media.

In cinema trailer the trend of MOS_{av} is slightly different. Unlike video clip, under 75 Kbps total bit rate, the best MOS_{av} is obtained with 16 Kbps audio bit

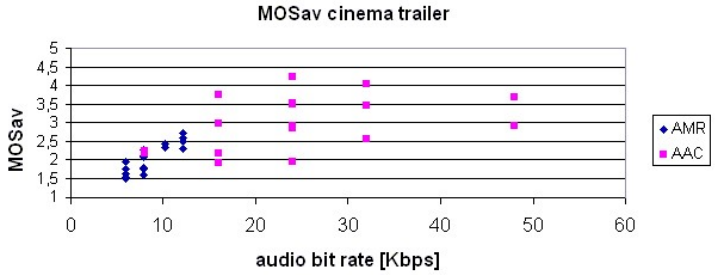

Figure 7.7: MOSav obtained through the use of AMR and AAC codec for cinema trailer.

rate and 59 Kbps video bit rate. Therefore it is possible to get a good result even with only 16 Kbps for audio. In this case the audio media is indeed simpler than in video clip and it is only an accompaniment to the scenes, so a lower audio bit rate than for video clip is enough. Even using 8, 16 or 24 Kbps for audio, the difference on MOS_{av} is not so great (see 56 Kbps total bit rate). As we expected, the influence of the audio quality on the audiovisual quality is greater for video clip than for cinema trailer.

If we consider the results of our subjective test for video clip and cinema trailer together, we note that the trend of MOS_{av} of our data in dependence of MOS_a and MOS_v is good described by a paraboloid, as for video call. The model for video clip and cinema trailer expressed in equation 7.1 is visualized in Figure 7.10.

The parabolic curve describes very well the mutually compensatory property between audio and video media. This model is designed using both AMR and AAC codecs. But AMR is not in general a suitable codec to be used for cinema trailer and video clip, as already explained above. The results obtained with AMR are all very low. So they cause a decreasing in the correlation of our model. In Figure 7.11 we can note the relation between the real MOS_{av} and the modelled MOS_{av}. This relation is quite linear: so a measured MOS_{av} value corresponds in

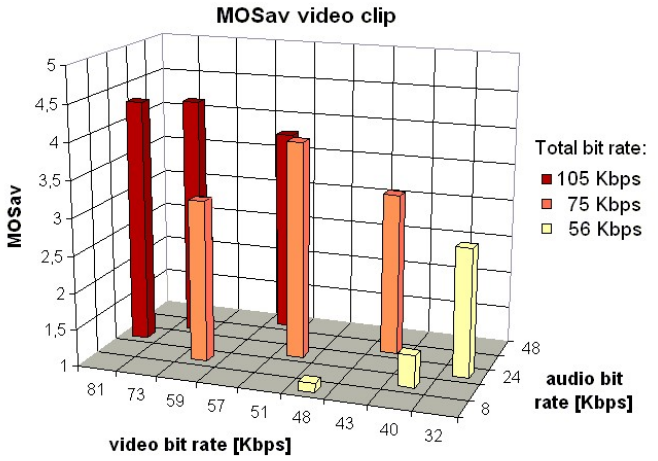

Figure 7.8: MOSav as function of audio and video bit rate, using Mpeg4 and AAC codecs.

general to an only modelled MOS_{av} value. That is true for the values obtained with AAC. For AMR instead, more values of modelled MOS_{av} correspond to a measured MOS_{av} value. This fact shows that our model suits better for AAC codec.

7.3 Conclusion

Our work centered on the subjective tests and on the video-only, audio-only and audiovisual metrics for the quality prediction.
The subjective tests were carried out using ACR method and displaying the sequences on an UMTS terminal. The test material was composed of sequences encoded with MPEG-4 or H.263 video codecs, combined with AAC or AMR audio codecs. We simulated several net conditions utilizing total bit rate from 56 kbps up to 105 kbps, video bit rate from 32 kbps up to 99 kbps and audio bit rate from 5,9 kbps up to 48 Kbps.

The audio and video models are based on several objective parameters. Some of these have their original idees in previous works on audio and video quality, others are completely new. In the metric design we chose to create two separate models

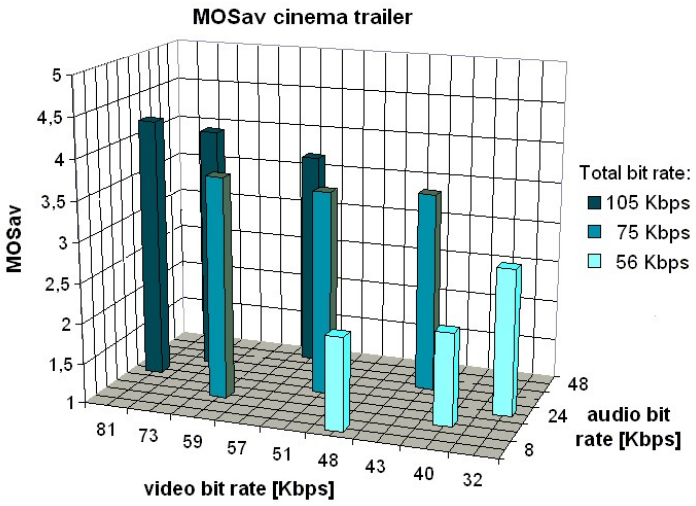

Figure 7.9: MOSav as function of audio and video bit rate, using Mpeg4 and AAC codecs.

for speech and music, since these two audio materials are differently perceived by human's ear and so differently evaluated. The effectiveness of these models is proved by the tests results.

The audio quality, the video quality, and the information entropy are factors of primary importance to determine the subjective audiovisual quality. Audio and video quality compensate often each other in the audiovisual quality. The models that describe at the best this property have parabolic form. The audiovisual model which represents the best trade-off between complexity and quality, has over 90% correlation with the real tests results. Now it would be interesting to determine the relation between information entropy and subjective judgment.

Although we utilized two video codecs and two audio codecs in our tests, this work centered on H.263 and AMR codecs.
We tested H.263 and Mpeg4 video codecs. They obtain generally the same results in the video quality. Nevertheless in some scenarios H.263 is not the most suitable codec. According our tests, MPEG-4 operates very better than H.263 in the video clip (Figure 7.5). In this clip the scenes are dark and change very fast: H.263

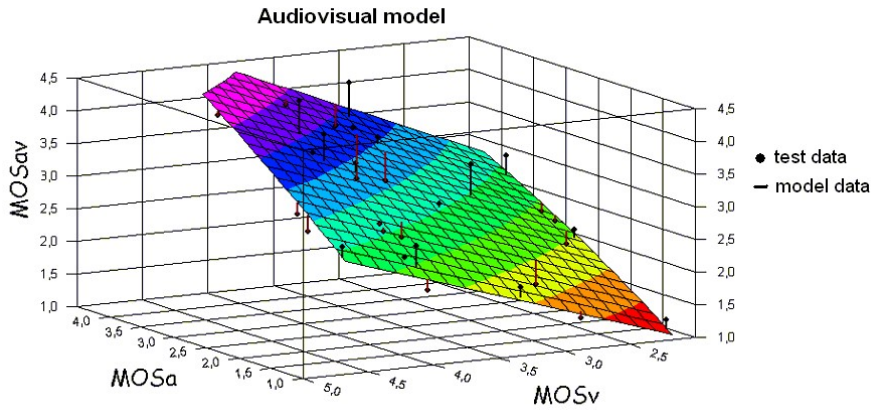

Figure 7.10: Video clip/Cinema trailer model.

allows to reach a video quality always smaller than 2.5 in a 5 grade MOS scale. On the contrary, in the video call and cinema trailer H.263 give results slightly higher than MPEG-4. Therefore it is not possible to determine a real winner between H.263 and MPEG-4 from our study.

AMR is an adaptive speech codec, which adapts its error protection level to the local radio channel and traffic conditions. In UMTS the AMR bit rates are controlled by the radio access network and do not depend on the speech activity. In our subjective tests we have utilized 5.9, 7.9, 10.2 and 12.2 Kbps as AMR bit rate, according to typical configurations of UMTS packet switched streaming services.

AMR was designed as speech codec in order to achieve an improved standard of voice quality and greater capacity. Therefore, as our tests results show, this codec suits very well for *video call*. In this case AMR codec represents *the best way to utilize the resources guaranteing a good subjective audiovisual quality*. It allows indeed to save a lot of bits maintaining a mean MOS_{av} around 3 that corresponds to a fair quality. Moreover the difference in the quality evaluation given by the use of 5.9, 7.9, 10.2 Kbps is not great. Higher is instead the MOS_{av} obtained with 12.2 Kbps. So it is convenient to utilize AMR codec at 12.2 Kbps for video call, because this bit rate produces a quality significantly higher than the others.

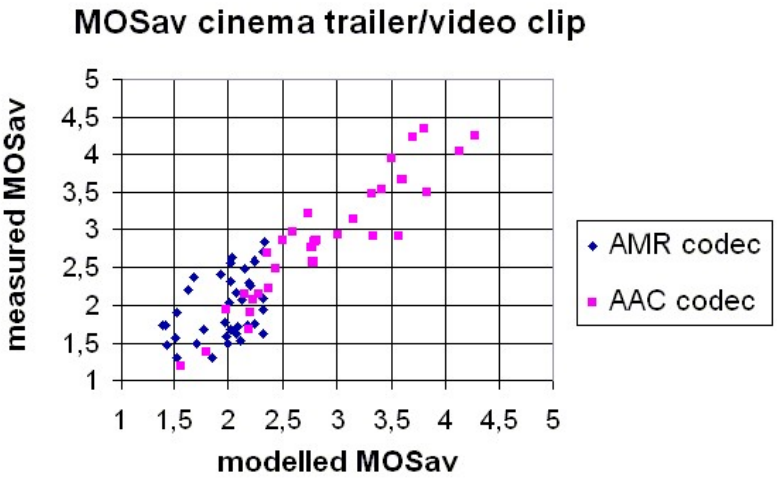

Figure 7.11: MOSav obtained by subjective tests (measured MOSav) versus MOSav obtained by our model (modelled MOSav).

We have tested AMR codec even on music (on video clip and cinema trailer). The results are in general very bad. The subjective MOS_{av} is always below 2.8. That means that the subjective audiovisual quality is poor in each case. *The use of AMR in a stream with music does not in general satisfy customs, so it makes no sense to use AMR for music coding.*

APPENDIX A:
Matlab implementations

In this appendix we show the Matlab implementations that we have elaborated for the objective parameters used in our metrics.

All the parameters are divided in three groups. The first group are speech parameters: they indicate the Auditory Distance (AD) between original and degraded signals. They have different implementations for AMR and AAC codecs.
The second parameters have been utilized for the music metric: they are the disturbance indicators A_ind and D_ind, and the distance indicator IFD.
The last parameters are instead necessary for the video metric. They investigate the spatial activity in the original and degraded video: they are Sigain, Siloss, Hvgain and Hvloss parameters.

The implementation of each group of parameters includes several Matlab functions. It follows the list of these functions (files .m):

Speech Parameters:

- AD_AMR.m
- AD_AAC.m

File AD_AMR contains the calculation of AD parameter for AMR codec.
File AD_AAC contains the calculation of AD parameter for AAC codec.

Music Parameters:

- main.m
- envelope.m
- FineTimeAlign2.m
- PowerRep2.m
- intensity_warping_of.m
- pseudo_Lp3.m
- Lpq_weight.m

The Figure 7.12 shows how the files work together

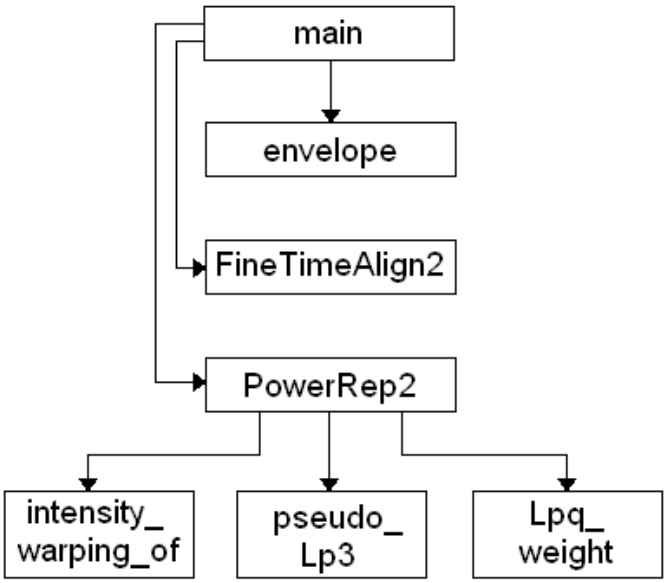

Figure 7.12: Block-diagram of the Matlab functions. Each function calls those indicated with the arrows.

Video Parameters:

- general.m

- region.m
- feature.m

The Figure 7.13 shows how the files work together.

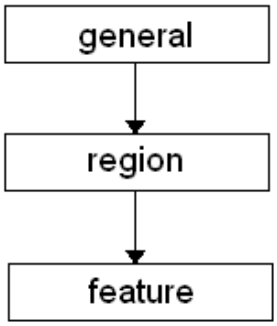

Figure 7.13: Block-diagram of the Matlab functions. Each function calls those indicated with the arrows.

Speech Parameters: file AD_AMR.m

```
clear all;
%THIS PROGRAM CALCULATES AD PARAMETER FOR AMR CODEC

% A. Signal Preparation
% reading samples to vectors x and y
% original signal is read to x and degragaded signal to y
%Here the tests signal are the original video call (vc_orig) and the
%degraded video call encoded with 5,9 Kbps. Both signals must be in WAV
%format.
[x, Fs, kbitx] =
wavread('C:\MATLAB7\work\SpeechParameters\vc_orig') [y, Fs, kbity]
= wavread('C:\MATLAB7\work\SpeechParameters\vc_AMR_5,9');
%AD algorithm requires that x and y are synchronized and have te same
%lenght.
%So I must synchronize the original and degraded signals:
  if length(x)<=length(y),
      k= length(x);
  else,
      k= length(y);
  end;

[XCF,Lags] = crosscorr(x, y, k-1);
%XCIF and Lags are vectors of 2k-1 elements. Lags contains the indices
%of the correlation between -(k-1) and +(k-1)
M=max(XCF)
  for r=1:(2*k-1),
         if XCF(r)>=M,
            delay=Lags(r);
         end;
```

```
  end;
%delay between original and degraded signals is where the crosscorrelation
%is maximum
%y follows x for "delay" samples
oldnx = length(x); oldny = length(y);
y(1:delay)=[]; % synchronizing samples
ny= length(y);
x((ny+1):oldnx)=[]; %Now y and x have the same length.
nx= length(x);

% plotting samples from the beginning and from the end of the original and
% degragaded signal
samples=10000; t=1:1:samples; figure (2); subplot(4,1,1); plot(x);
axis ([0 samples 1.1*min(x) 1.1*max(x)]); title('Start of the
original sample'); xlabel('index'); subplot(4,1,2); plot(y); axis
([0 samples 1.1*min(y) 1.1*max(y)]); title('Start of the
degragaded sample'); xlabel('index'); subplot(4,1,3); plot(x);
axis ([nx-samples nx 1.1*min(x) 1.1*max(x)]); title('End of the
original sample'); xlabel('index'); subplot(4,1,4); plot(y); axis
([ny-samples ny 1.1*min(y) 1.1*max(y)]); title('End of the
degragaded sample'); xlabel('index');

% removing mean value from each term of x and y
mean_x=mean(x);   %mean value of x
mean_y=mean(y);   %mean value of y
%mean value is removed from each x and y component
for i=1:n1
    x(i)=x(i)-mean_x;
    y(i)=y(i)-mean_y;
end
```

```
% x and y are normalized to a common RMS level
rms_x=norm(x)/sqrt(n1);  %rms of x
rms_y=norm(y)/sqrt(n1);  %rms of y
%normalization
for i=1:n1
    x(i)=x(i)/rms_x;
    y(i)=y(i)/rms_y;
end

% B. Transformation to Frequency Domain
%x and y are broken in to series of frames, 128 samples each frame and
%frame overlap 50%, any samples beyond the final full frame are discarded
[xbuf1,zx]=buffer(x,128,64);    %xbuf1 includes only full frames, zx may
                                %include samlpes beyond the final full
                                %frame if they exist
[ybuf1,zy]=buffer(y,128,64);
% each frame is multiplied with hamming window
frames=size(xbuf1,2);  %number of frames
n2=frames; for j=1:n2
    for i=1:128
       xbuf(i,j)=xbuf1(i,j)*(0.54-0.46*cos((2*pi*(i-1))/127));
       ybuf(i,j)=ybuf1(i,j)*(0.54-0.46*cos((2*pi*(i-1))/127));
    end
end
%fft for xbuf and ybuf
xfft=abs(fft(xbuf,128)); yfft=abs(fft(ybuf,128));
% only squared magnitudes of samples 1-65 from each frame are retained.
% these are stored in matrices X and Y, which both are 65*n2 matrices
% (n2=number of frames)
for j=1:n2
```

```
    for i=1:65
        X(i,j)=xfft(i,j)^2;
        Y(i,j)=yfft(i,j)^2;
    end
end

% C. Frame Selection
%only frames that exceed energy threshold in both X and Y are used in
%calculating AD
%vectors for energies of frames
xenergy=zeros(1,n2); yenergy=zeros(1,n2); for j=1:n2
    for i=1:65
        xenergy(j)=xenergy(j)+X(i,j);
        yenergy(j)=yenergy(j)+Y(i,j);
    end
end
%thresholds
xthreshold=10^(-1.5)*max(xenergy);
ythreshold=10^(-3.5)*max(yenergy);
%frames that meet both of these energy thresholds are retained
%if any frame contains one or more samples that are equal to zero, that
%frame is eliminated from both X and Y
%helps with detecting zeros, 0 means there is no zeros in this frame and 1
%means that there is a zero in this frame of either X or Y
iszero=zeros(1,n2); for j=1:n2
    for i=1:65
        if ((X(i,j)==0) || (Y(i,j)==0))
            iszero(j) = 1;
        end
    end
```

```
end
j=1;    %frame of old X and Y matrices
k=1;    %frame of new matrices
% removing frames that do not meet the conditions mentioned above
while j <= n2
    if ((xenergy(j)>=xthreshold)&&(yenergy(j)>=ythreshold)&&iszero(j) == 0)
        for i=1:65
            X1(i,k)=X(i,j);
            Y1(i,k)=Y(i,j);
        end
        k=k+1;
        j=j+1;
    else
        j=j+1;
    end
end
n3=size(X1,2); %number of frames after removing extra frames
if (n3>0)
    % renaming matrices
    X=X1;
    Y=Y1;
end;


  % D. Perceived Loudness Approximation
    %transformation into dB
    for j=1:n3
        for i=1:65
            X(i,j)=10*log10(X(i,j));
            Y(i,j)=10*log10(Y(i,j));
        end
```

```
    end


  % E.Frequency Measuring Normalizing Block
    f1=zeros(1,65);
    f2=zeros(1,65);
    for i=1:65
        meanX=0;
        meanY=0;
        for j=1:n3
            meanX=meanX+X(i,j);
            meanY=meanY+Y(i,j);
        end
        meanX=meanX/n3;
        meanY=meanY/n3;
        f1(i)= meanY-meanX;
    end
    %normalizing Y
    for j=1:n3
        for i=1:65
            Y(i,j)=Y(i,j)-f1(i);
        end
    end
    %determining f2
    for i=1:65
        f2(1,i)=f1(1,i)-f1(1,17);   %normalizes measurement to 1kHz
    end
    %determining f3
    f3=zeros(1,16);
    for i=1:16
        sumf3=0;
```

```
        for j=1:4
            sumf3=sumf3+f2(1+4*(i-1)+j);    %makes the measurement smooth
        end
        f3(i)=sumf3/4;
    end
    %saving measurements in vector m
    m=zeros(12,1);
    m(1)=f3(1);
    m(2)=f3(2);
    m(3)=f3(13);
    m(4)=f3(14);

% F. Structure 1: Time Measuring Normalizing Block
    % 1) Largest scale TMN
    t0=zeros(1,n3);
    for j=1:n3
        for i=2:65
            t0(j)=t0(j)+Y(i,j)-X(i,j);
        end
        t0(j)=t0(j)/64;
    end
    % normalizing Y
    for j=1:n3
        for i=2:65
            Y(i,j)=Y(i,j)-t0(j);
        end
    end
    % saving positive portion of measurement
    summ5=0;
    for j=1:n3
```

```
    if (t0(j)>0)
    summ5=summ5+t0(j);
    end
end
m(5)=summ5/n3;

% 2) defining vector for band limits
g=[2 7 12 19 29 43 66]';
%implementing six small scale TMNB's
for k=1:6;
    t1=zeros(1,n3);
    for j=1:n3
        Ysum=0;
        Xsum=0;
        h=(1/(g(k+1)-g(k)));    %just a multiplicant
            for i=g(k):(g(k+1)-1);
                Ysum=Ysum+Y(i,j);
                Xsum=Xsum+X(i,j);
            end
            Ysum=Ysum*h;
            Xsum=Xsum*h;
        t1(j)=Ysum-Xsum;
    end
% normalizing Y
    for j=1:n3
        for i=g(k):(g(k+1)-1)
            Y(i,j)=Y(i,j)-t1(j);
        end
    end
% saving positive portion of measurement
```

```
    sum=0;
    for j=1:n3
        if (t1(j)>0)
        sum=sum+t1(j);
        end
    end
    m(5+k)=sum/n3;
end

% 3) residual measurement
for j=1:n3
    for i=1:65
        t2(i,j)=Y(i,j)-X(i,j);
    end
end
% saving positive portion of residual measurement
sum=0;
for j=1:n3
    for i=2:65
        if (t2(i,j)>0)
        sum=sum+t2(i,j);
        end
    end
end
m(12)=sum/(n3*64);

% H. Linear Combinations and Logistic Functions
% defining vectors w1 and w2 for structures 1 and 2 and logistic
% parametres a1, b1, b1, b2
w1=[0.0034 -0.0650 -0.1304 0.1352 0.5931 0.2040 0.5577 0.1008 0.0627
```

```
        0.0052 0.0107 1.1037];
    w2=[0.0000 -0.0837 -0.1199 0.1260 0.1660 0.6387 0.2195 0.0122 1.5544
        0.0954 0.1720];
    a1=1.0000;
    b1=-4.6877;
    a2=1.0000;
    b2=-3.0613;
    % AD value
    AD=w1*m
    %final algorithm output
    L=1/(1+exp(a1*AD+b1))
    else
   AD=('Input vectors do not contain suitable signals for this algorithm.')
end
```

Speech Parameters: file AD_AAC.m

```
clear all;
%THIS PROGRAM CALCULATES AD PARAMETER FOR AAC CODEC

% A. Signal Preparation
% reading samples to vectors x and y
% original signal is read to x and degragaded signal to y
%Here the tests signal are the original video call (vc_orig) and the
%degraded video call encoded with 32 Kbps. Both signals must be in WAV
%format.
[x, Fs, kbitx] =
wavread('C:\MATLAB7\work\SpeechParameters\vc_orig'); [y, Fs,
kbity] = wavread('C:\MATLAB7\work\SpeechParameters\vc_AAC_32');
%AD algorithm requires that x and y are synchronized and have te same
%lenght.
%So I must synchronize the original and degraded signals:
if length(x)<=length(y),
      k= length(x);
  else,
      k= length(y);
end;

[XCF,Lags] = crosscorr(x, y, k-1);
%XCF and Lags are vectors of 2k-1 elements. Lags contains the indices
%of the correlation between -(k-1) and +(k-1)
M=max(XCF)
  for r=1:(2*k-1),
         if XCF(r)>=M,
            delay=Lags(r);
         end;
```

```
  end;
%delay between original and degraded signals is where the crosscorrelation
%is maximum
nx = length(x);
y(1:delay)=[]; % synchronizing samples removing first delay samples.
               % But y is longer than x.
oldny = length(y);
y((nx+1):oldny)=[]; %Now y and x have the same length.
ny= length(y);

% plotting samples from the beginning and from the end of the original and
% degragaded signal
samples=10000; t=1:1:samples; figure (2); subplot(4,1,1); plot(x);
axis ([0 samples 1.1*min(x) 1.1*max(x)]); title('Start of the
original sample'); xlabel('index'); subplot(4,1,2); plot(y); axis
([0 samples 1.1*min(y) 1.1*max(y)]); title('Start of the
degragaded sample'); xlabel('index'); subplot(4,1,3); plot(x);
axis ([nx-samples nx 1.1*min(x) 1.1*max(x)]); title('End of the
original sample'); xlabel('index'); subplot(4,1,4); plot(y); axis
([ny-samples ny 1.1*min(y) 1.1*max(y)]); title('End of the
degragaded sample'); xlabel('index');

% removing mean value from each term of x and y
mean_x=mean(x);   %mean value of x
mean_y=mean(y);   %mean value of y
%mean value is removed from each x and y component
for i=1:n1
    x(i)=x(i)-mean_x;
    y(i)=y(i)-mean_y;
end
```

```
% x and y are normalized to a common RMS level
rms_x=norm(x)/sqrt(n1);  %rms of x
rms_y=norm(y)/sqrt(n1);  %rms of y
%normalization
for i=1:n1
    x(i)=x(i)/rms_x;
    y(i)=y(i)/rms_y;
end

% B. Transformation to Frequency Domain
%x and y are broken in to series of frames, 128 samples each frame and
%frame overlap 50%, any samples beyond the final full frame are discarded
[xbuf1,zx]=buffer(x,128,64);    %xbuf1 includes only full frames, zx may
                                %include samlpes beyond the final full
                                %frame if they exist
[ybuf1,zy]=buffer(y,128,64);
frames=size(xbuf1,2);  %number of frames
n2=frames;
% each frame is multiplied with hamming window
for j=1:n2
    for i=1:128
       xbuf(i,j)=xbuf1(i,j)*(0.54-0.46*cos((2*pi*(i-1))/127));
       ybuf(i,j)=ybuf1(i,j)*(0.54-0.46*cos((2*pi*(i-1))/127));
    end
end
%fft for xbuf and ybuf
xfft=abs(fft(xbuf,128)); yfft=abs(fft(ybuf,128));
% only squared magnitudes of samples 1-65 from each frame are retained.
% these are stored in matrices X and Y, which both are 65*n2 matrices
% (n2=number of frames)
```

```
for j=1:n2
    for i=1:65
        X(i,j)=xfft(i,j)^2;
        Y(i,j)=yfft(i,j)^2;
    end
end

% C. Frame Selection
%only frames that exceed energy threshold in both X and Y are used in
%calculating AD
%vectors for energies of frames:
xenergy=zeros(1,n2); yenergy=zeros(1,n2); for j=1:n2
    for i=1:65
        xenergy(j)=xenergy(j)+X(i,j);
        yenergy(j)=yenergy(j)+Y(i,j);
    end
end
%thresholds
xthreshold=10^(-1.5)*max(xenergy);
ythreshold=10^(-3.5)*max(yenergy);
%frames that meet both of these energy thresholds are retained
%if any frame contains one or more samples that are equal to zero, that
%frame is eliminated from both X and Y
%helps with detecting zeros, 0 means there is no zeros in this frame and 1
%means that there is a zero in this frame of either X or Y
iszero=zeros(1,n2); for j=1:n2
    for i=1:65
        if ((X(i,j)==0) || (Y(i,j)==0))
            iszero(j) = 1;
        end
```

```
    end
end
j=1;    %frame of old X and Y matrices
k=1;    %frame of new matrices
% removing frames that do not meet the conditions mentioned above
while j <= n2
    if ((xenergy(j)>=xthreshold)&&(yenergy(j)>=ythreshold)&&iszero(j) == 0)
        for i=1:65
            X1(i,k)=X(i,j);
            Y1(i,k)=Y(i,j);
        end
        k=k+1;
        j=j+1;
    else
        j=j+1;
    end
end
n3=size(X1,2); %number of frames after removing extra frames
if (n3>0)
    % renaming matrices
    X=X1;
    Y=Y1;
end;

% D. Perceived Loudness Approximation
    %transformation into dB
    for j=1:n3
        for i=1:65
            X(i,j)=10*log10(X(i,j));
            Y(i,j)=10*log10(Y(i,j));
```

```
        end
    end

% E.Frequency Measuring Normalizing Block
    f1=zeros(1,65);
    f2=zeros(1,65);
    for i=1:65
        meanX=0;
        meanY=0;
        for j=1:n3
            meanX=meanX+X(i,j);
            meanY=meanY+Y(i,j);
        end
        meanX=meanX/n3;
        meanY=meanY/n3;
        f1(i)= meanY-meanX;
    end
    %normalizing Y
    for j=1:n3
        for i=1:65
            Y(i,j)=Y(i,j)-f1(i);
        end
    end
    %determining f2
    for i=1:65
        f2(1,i)=f1(1,i)-f1(1,17);   %normalizes measurement to 1kHz
    end
    %determining f3
    f3=zeros(1,16);
    for i=1:16
```

```
        sumf3=0;
        for j=1:4
            sumf3=sumf3+f2(1+4*(i-1)+j);    %makes the measurement smooth
        end
        f3(i)=sumf3/4;
    end
    %saving measurements in vector m
    m=zeros(12,1);
    m(1)=f3(1);
    m(2)=f3(2);
    m(3)=f3(13);
    m(4)=f3(14);

% F. Structure 1: Time Measuring Normalizing Block
   % 1) Largest scale TMN
    t0=zeros(1,n3);
    for j=1:n3
        for i=2:65
            t0(j)=t0(j)+Y(i,j)-X(i,j);
        end
        t0(j)=t0(j)/64;
    end
    % normalizing Y
    for j=1:n3
        for i=2:65
            Y(i,j)=Y(i,j)-t0(j);
        end
    end
    % saving positive portion of measurement
    summ5=0;
```

```
for j=1:n3
    if (t0(j)>0)
    summ5=summ5+t0(j);
    end
end
m(5)=summ5/n3;

% 2) defining vector for band limits
g=[2 7 12 19 29 43 66]';
%implementing six small scale TMNB's
for k=1:6;
    t1=zeros(1,n3);
    for j=1:n3
        Ysum=0;
        Xsum=0;
        h=(1/(g(k+1)-g(k)));    %just a multiplicant
            for i=g(k):(g(k+1)-1);
                Ysum=Ysum+Y(i,j);
                Xsum=Xsum+X(i,j);
            end
            Ysum=Ysum*h;
            Xsum=Xsum*h;
        t1(j)=Ysum-Xsum;
    end
% normalizing Y
    for j=1:n3
        for i=g(k):(g(k+1)-1)
            Y(i,j)=Y(i,j)-t1(j);
        end
    end
```

```
    % saving positive portion of measurement
        sum=0;
        for j=1:n3
            if (t1(j)>0)
            sum=sum+t1(j);
            end
        end
        m(5+k)=sum/n3;
    end

    % 3) residual measurement
   for j=1:n3
        for i=1:65
            t2(i,j)=Y(i,j)-X(i,j);
        end
    end
    % saving positive portion of residual measurement
    sum=0;
    for j=1:n3
        for i=2:65
            if (t2(i,j)>0)
            sum=sum+t2(i,j);
            end
        end
    end
    m(12)=sum/(n3*64);

% H. Linear Combinations and Logistic Functions
    % defining vectors w1 and w2 for structures 1 and 2 and logistic
    % parametres a1, b1, b1, b2
```

```
    w1=[0.0034 -0.0650 -0.1304 0.1352 0.5931 0.2040 0.5577 0.1008 0.0627
        0.0052 0.0107 1.1037];
    w2=[0.0000 -0.0837 -0.1199 0.1260 0.1660 0.6387 0.2195 0.0122 1.5544
        0.0954 0.1720];
    a1=1.0000;
    b1=-4.6877;
    a2=1.0000;
    b2=-3.0613;
    % AD value
    AD=w1*m
    %final algorithm output
    L=1/(1+exp(a1*AD+b1))
    else
   AD=('Input vectors do not contain suitable signals for this algorithm.')
end
```

Music Parameters: file main.m

```
function [A_ind, D_ind, IFD]=main
%clear all;

%The degraded and original signals are read respectively in deg and orig
%vectors. Both signals must be in WAV format.
%Here the original signal is the video clip (vm_orig). The degraded one is the
%video clip encoded with 32 Kbps.
deg = wavread('C:\programmi\MATLAB7\work\MusicParameters\vm_AAC_32');
orig = wavread('C:\programmi\MATLAB7\work\MusicParameters\vm_orig');
Norig = wavread('C:\programmi\MATLAB7\work\MusicParameters\vm_orig','size');
%N is the number of input samples
Nsamplesorig= Norig(1,1);
Ndeg = wavread('C:\programmi\MATLAB7\work\MusicParameters\vm_AAC_32','size');
Nsamplesdeg=Ndeg(1,1);
if Nsamplesdeg>=Nsamplesorig,
    Nsamples=Nsamplesorig;
else,
    Nsamples=Nsamplesdeg;
end;
xs= orig;
ys= deg;

%Crude delay algorithm in order to find the delay between original and
%degraded signal.
[xes, yes, d_env]=envelope(Nsamples, xs, ys);
%d_env is the crude delay between original and degraded audio

%Fine delay algorithm in order to find a more precise delay between
%original and degraded signal.
```

```
[fin_delay]=FineTimeAlign2(xes, yes, d_env);

%Calculation of our three parameters
[A_ind, D_ind, IFD]=tomPowerRep2(xs, ys, fin_delay);

end
```

Music Parameters: file envelope.m

```
function [xout, yout, delay]=envelope(N, x, y)
%The frames  are formed by 64 samples without overlapping
[ybuf,zy]=buffer(y,64);
[xbuf,zx]=buffer(x,64);
ky=size(ybuf, 2);
kx=size(xbuf, 2);
%The crude delay is calculated on the cross-correlation of original and
%degraded envelopes.
enx=zeros(kx, 1);
eny=zeros(ky, 1);
%Xfbuf=fft(xbuf,64);
%Yfbuf=fft(ybuf,64);
  for j=1:kx,
     enx(j)=sum(xbuf(:,j).^2);          %enx is the energy of a frame
     env_x(j)= log10(max(enx(j), 1));  %env_x is the envelope of each frame
  end;
  for j=1:ky,
     eny(j)=sum(ybuf(:,j).^2);
     env_y(j)= log10(max(eny(j), 1));
  end;

if kx>=ky,
```

```
     k=ky;
else,
     k=kx;
end;
[XCF,Lags] = crosscorr(env_x, env_y, k-1);
%XCIF and Lags are vectors of 2k-1 elements. Lags contains the indices
%of the correlation between -(k-1) and +(k-1)
M=max(XCF)
  for r=1:(2*k-1),
         if XCF(r)>=M,
            delay=Lags(r); %delay is where the maximum of the
                           %cross-correlation is.
         end;
  end;
%a positive delay means that y comes later than x
%a negative delay means that x comes later than y
%Now I apply envelope-based alignment: I must cancel the delay of yout
yout=zeros(N,1);
if delay==0,
    yout= y;
elseif delay>0,
   y(1:delay)=[]; %So y becomes long N-delay samples
   yout(1:(N-delay))=y;
else,
    y((N+delay+1):N)=[];
    yout((-delay+1):N)= y(:);
end;
%yout is long N-delay samples and it´s aligned with the original xout
xout= x;
end
```

Music Parameters: file FineTimeAlign2.m

```
function [delay]=FineTimeAlign2(x, y, d)
%x and y are broken in to series of frames, 1024 samples each frame and
%frame overlap 75%(i.e.768 samples)
[xbuf,zx]=buffer(x,1024,768);
[ybuf,zy]=buffer(y,1024,768);
%each frame is multiplied with Hann window
NumFrx=size(xbuf, 2);
NumFry=size(ybuf, 2);
  for f=1:NumFrx,
    for w=1:1024,
      xbufH(w,f)=xbuf(w,f)*(0.5*(1-cos(2*pi*(w-1)/1023)));
    end;
  end;
  for fy=1:NumFry,
    for wy=1:1024,
      ybufH(wy,fy)=ybuf(wy,fy)*(0.5*(1-cos(2*pi*(wy-1)/1023)));
    end;
  end;
if NumFry<=NumFrx,
     NumFr=NumFry;
else,
     NumFr=NumFrx;
end;
  for c=1:NumFr,
     [XCF,Lags] = crosscorr(xbufH(:,c), ybufH(:,c), 1023);
     %XCIF and Lags are vectors of 2047 elements. Lags contains the indices
     %of the correlation between -1023 and +1023
      M=max(XCF);
         for r=1:2047,
```

```
            if XCF(r)>=M,
                d_frame=Lags(r);
            end;
        end;
    %d_frames is the index where there is the maximum of the correlation.
    %It indicates the delay for each frame
      if c==1,
          del=d_frame;
      else,
          del= vertcat(del, d_frame);
      end;
  end;
  %del is a vector of NumFr elements. It contains the delays for every
  %frame

%Now I build an histogram of these delays.
hist=zeros(1024,2);
% In the first column of hist I put the delay.
% In the second column of hist I put the number of frames with that delay.
i=1;
  for j=0:1023,    %I must investigate also the delay 0.
       for c=1:NumFr,
           if del(c)==j,
               hist(j+1,1)=j;
               hist(j+1,2)=i;  %hist is the not-smoothed histogram
               i=i+1;
           end;
       end;
       i=1;
  end;
```

```
%The histogram is then smoothed by convolution with a symmetric
%triangular kernel.
tri= [0; 0.3; 0.6; 0.9; 0.9; 0.6; 0.3; 0]; %triangular kernel
colhist=hist(:,2);
h=conv(tri, colhist); %h is long 1031 samples. It is simmetrical. I ignore
                      %the first 4 and the last 3 samples
h(1:4)=[];
h(1025:1027)=[];
hist_smooth(:,1)=hist(:,1); %the first column of the histogram remain the
                            %same (no convolution)
hist_smooth(:,2)=h;
%hist_smooth is the smoothed histogram
%The index of the maximum in the smoothed histogram is the fine delay
%(d_hist)
M2=max(hist_smooth(:,2));
   for z=1:1024,
             if hist_smooth(z,2)>=M2,
                d_hist =hist_smooth(z,1);
             end;
   end;
%The final delay is the sum of crude delay and fine delay
delay =d_hist+ d;
end
```

Music Parameters: file PowerRep2.m

```
function[A_indicator, D_indicator, FreqDistance]=PowerRep2(xir, yir, delay)
D_POW_F= 2;
D_POW_S= 6;
D_POW_T= 2;
A_POW_F= 1;
```

```
A_POW_S= 6;
A_POW_T= 2;
D_WEIGHT= 1;
A_WEIGHT= 0,309;
SP= 6.910853e-6;
%x and y are broken into series of frames, 512 samples each frame and
%frame overlap 50%(i.e. 256 samples)
[xbuf,zx]=buffer(xir,512,256);
[ybuf,zy]=buffer(yir,512,256);
Numy=size(ybuf, 2);
Numx=size(xbuf, 2);
if Numx>=Numy,
    Num=Numy;
 else,
    Num=Numx;
end;

%each frame is multiplied with Hann window
  for f=1:Num,
    for w=1:512,
      xbufH(w,f)=xbuf(w,f)*(0.5*(1-cos(2*pi*(w-1)/511)));
      ybufH(w,f)=ybuf(w,f)*(0.5*(1-cos(2*pi*(w-1-delay)/511)));
      %the start points of the windows in the degraded signal are shifted
      %over the delay
    end;
  end;
FxbufH= fft(xbufH); %fft returns the Fourier transform of each column of
                    %the matrix xbufH
FybufH= fft(ybufH);
AbsFxbufH= vabs(FxbufH);
```

```
AbsFybufH= vabs(FybufH);
PXwirss= AbsFxbufH.^2;
PYwirss= AbsFybufH.^2;

%Now the frequency axis is warped and converted to a Bark frequency scale.
%Bark scale
%Each Bark band contains more frequency-samples.
%In nrhz it is written the number of samples for each Bark band. For
%example the first Bark band contains 2 samples.
nrhz=[2; 2; 2; 2; 2; 2; 2; 2; 4; 2; 2; 2; 2; 2;4;2;2;4;4;4;4;4;4;4;6;6;6;
    6;8;6;8;10;8;10;12;12;14;16;18;18;24;24;30;32;36;42;50;40];
%power density correction function for each Bark band:
corr_fact=[100.000000; 99.999992; 100.000000; 100.000008; 100.000008;
        100.000015; 99.999992; 99.999969; 50.000027; 100.000000; 99.999969;
        100.000015; 99.999947; 100.000061; 53.047077;110.000046;117.991989;
        65.000000; 68.760147; 69.999931; 71.428818; 75.000038; 76.843384;
          80.968781; 88.646126; 63.864388; 68.155350; 72.547775; 75.584831;
          58.379192; 80.950836; 64.135651; 54.384785; 73.821884; 64.437073;
          59.176456; 65.521278; 61.399822; 58.144047; 57.004543; 64.126297;
          54.311001; 61.114979; 55.077751; 56.849335; 55.628868; 53.137054;
          54.985844;  79.546974];
%Constuction of pitch power densities
PPX=zeros(49,Num);
PPY=zeros(49,Num);
   for c=1:Num,
       colx=PXwirss(:,c);
       coly=PYwirss(:,c);
       for i=1:49,
           x=colx(1:nrhz(i));    %x is a Bark band
           y=coly(1:nrhz(i));    %y is a Bark band
```

```
            %here I am in a Bark band in a frame
            sx=sum(x)*SP;
            sy=sum(y)*SP;
            sx_corr=sx* corr_fact(i);
            sy_corr=sy* corr_fact(i);
            PPX(i,c)=sx_corr;
            PPY(i,c)=sy_corr;
        end;
    end;

%Finding silent frames
thresh_power=[51286152.000000; 2454709.500000; 70794.593750; 4897.788574;
              1174.897705;     389.045166;     104.712860;     45.708820;
              17.782795;       9.772372;         4.897789;     3.090296;
              1.905461;        1.258925;         0.977237;      0.724436;
              0.562341;        0.457088;          0.389045;     0.331131;
                0.295121;      0.269153;          0.257040;     0.251189;
                0.251189;       0.251189;         0.251189;     0.263027;
                0.288403;       0.309030;         0.338844;     0.371535;
                0.398107;       0.436516;         0.467735;     0.489779;
                0.501187;       0.501187;         0.512861;     0.524807;
                0.524807;       0.524807;         0.512861;     0.478630;
                0.426580;       0.371535;         0.363078;     0.416869;
                0.537032];
thresh= thresh_power*1e2;
  for j=1:Num,
      p=1;
      for r=1:49,
          if PPX(r,j)>thresh(r),
              over_threshx(p)=PPX(r,j);
```

```
              p=p+1;
           end;
       end;
        if p==1, %i.e if over_thresh is empty,
             vectx(j)=0,
        else,
             vectx(j)=sum(over_threshx);
        end;
  end;
%vectx is a row of Num elements
for jy=1:Num,
       p=1;
       for ry=1:49,
           if PPY(ry,jy)>thresh(ry),
               over_threshy(p)=PPY(ry,jy);
               p=p+1;
           end;
       end;
        if p==1, %i.e if over_thresh is empty,
             vecty(jy)=0,
        else,
             vecty(jy)=sum(over_threshy);
        end;
  end;
%vecty is a row of Num elements
%Silent frame:
silent_framex=zeros(1,Num); %row vector
for z=1:Num,
     if vectx(z)<1e7,
         silent_framex(z)=1;
```

```
     else,
         silent_framex(z)=1;
     end;
end;
%where there is 1, the frame is silent
%Silent frame:
silent_framey=zeros(1,Num); %row vector
for z=1:Num,
     if vecty(z)<1e7,
         silent_framey(z)=1;
     else,
         silent_framex(z)=1;
     end;
end;
%where there is 1, the frame is silent

%Partial compensation of the original pitch power density to equalize the
%original to the degraded signal.
%Construction of the compensaty factor
%addition per rows:
average_power_x= zeros(49,1); %column vector
  for row=1:49,
      p=1;
      for colu=1:Num,
          if (silent_framex(colu)==1) & (PPX(row,colu)>thresh(row)),
              %i.e. if the frame isn´t silent,
              no_silentx(p)=PPX(row,colu);
              p=p+1;
          end;
      end;
```

```
      if p~=1, %i.e. if the conditions above are been sotisfied at least
               %once, otherwise average_power_x remains zero
          average_power_x(row)=sum(no_silentx)/Num;
      end;
  end;
  %addition per rows:
average_power_y= zeros(49,1); %column vector
  for rowy=1:49,
      p=1;
      for coluy=1:Num,
          if (silent_framey(coluy)==1) & (PPY(rowy,coluy)>thresh(rowy)),
              %i.e. if the frame isn´t silent,
              no_silenty(p)=PPY(rowy,coluy);
              p=p+1;
          end;
      end;
      if p~=1, %i.e. if the conditions above are been sotisfied at least
               %once, otherwise average_power_x remains zero
          average_power_y(rowy)=sum(no_silenty)/Num;
      end;
  end;
  for b=1:49,
     x=(average_power_y(b)+1000)/(average_power_x(b)+1000)
     %x is the compensatory factor
      if x>100,
          x=100;
      end;
      if x<0.001,
          x=0.001;
      end;
```

```
      for r=1:Num,
          PPX_ratio(b,r)=PPX(b,r)*x; %this is the original power density
                                     %equalized to the degraded signal
      end;
  end;

%Partial compensation of the degraded pitch power density for short-term
%gain variations
  c=1;
  r=1;
  for c=1:Num,
      p=1;
      for r=1:49,
          if PPY(r,c)>thresh_power(r),
              over_thry(p)=PPY(r,c);
              p=p+1;
          end;
      end;
       if p==1, %i.e if over_thry is empty,
            total_audible_pow_deg(c)=0,
       else,
            total_audible_pow_deg(c)=sum(over_thry);
       end;
  end;
  c=1;
  r=1;
  for c=1:Num,
      p=1;
      for r=1:49,
          if PPX_ratio(r,c)>thresh_power(r),
```

```
              over_thrx(p)=PPX_ratio(r,c);
              p=p+1;
          end;
      end;
       if p==1, %i.e if over_thry is empty,
            total_audible_pow_ref(c)=0,
       else,
            total_audible_pow_ref(c)=sum(over_thrx);
       end;
  end;
scale=(total_audible_pow_ref+5e3)./(total_audible_pow_deg+5e3);
%scale is long Num
new_scale(1)=scale(1)*0.2*0.8;
i=2;
  for i=2:Num,
      new_scale(i)=new_scale(i-1)*0.2+ scale(i)*0.8;
  end;

j=1;
  for j=1:Num,
      if new_scale(j)>5,
          new_scale(j)=5;
      end;
       if new_scale(j)<3e-4,
          new_scale(j)=3e-4;
      end;
  end;
c=1;
  for c=1:Num,
      PPY_ratio(:,c)=PPY(:,c)*new_scale(c); %this is the compensated
```

```
                                                %distorted pitch power density
  end;

%Untill here we have buit PPX_ratio and PPY_ratio
%Now we build the loudness densities
centre_band_bark =[ 0.078672;     0.316341;     0.636559;     0.961246;
                    1.290450;      1.624217;     1.962597;     2.305636;
                    2.653383;     3.005889;      3.363201;     3.725371;
                    4.092449;     4.464486;      4.841533;     5.223642;
                    5.610866;     6.003256;     6.400869;     6.803755;
                    7.211971;     7.625571;     8.044611;     8.469146;
                    8.899232;     9.334927;     9.776288;     10.223374;
                    10.676242;     11.134952;   11.599563;     12.070135;
                    12.546731;     13.029408;     13.518232;   14.013264;
                    14.514566;     15.022202;     15.536238;     16.056736;
                   16.583761;     17.117382;     17.657663;     18.204674;
                   18.758478;     19.319147;     19.886751;     20.461355;
                   21.043034];
width_band_bark = [ 0.157344;     0.317994;     0.322441;     0.326934;
                    0.331474;     0.336061;     0.340697;     0.345381;
                    0.350114;     0.354897;     0.359729;     0.364611;
                    0.369544;     0.374529;     0.379565;     0.384653;
                    0.389794;     0.394989;     0.400236;     0.405538;
                    0.410894;     0.416306;     0.421773;     0.427297;
                    0.432877;     0.438514;     0.444209;     0.449962;
                    0.455774;     0.461645;     0.467577;     0.473569;
                    0.479621;     0.485736;     0.491912;     0.498151;
                    0.504454;     0.510819;     0.517250;     0.523745;
                    0.530308;     0.536934;     0.543629;     0.550390;
                    0.557220;     0.564119;     0.571085;     0.578125;
```

```
                    0.585232];
[LX]=intensity_warping_of(Num, centre_band_bark, PPX_ratio, thresh_power);
%LX is the original loudness density
[LY]=intensity_warping_of(Num, centre_band_bark, PPY_ratio, thresh_power);
%LY is the degraded loudness density
%Disturbance density:
disturbance=LY-LX;

%Aggregation of disturbance densities over frequency
%frame_disturance indicates disturbes per frame
[frame_disturbance]=pseudo_Lp3(Num, width_band_bark, disturbance);
ratio=(PPY_ratio+50)./(PPX_ratio+50);
H=ratio.^1.2; %H is the asymmetry factor
for j=1:Num,
    for i=1:49,
        disturbance(i,j)=disturbance(i,j)*H(i,j); %cell-wise multiplication
                                                  %with an asymetry factor
    end;
end;
%Now disturbance has an assymetrical factor. So one can obtain
%frame_disturbance_asym
[frame_disturbance_asym]=pseudo_Lp3(Num, width_band_bark, disturbance);
R= ((total_audible_pow_ref+ 1e5)./1e7).^0.04;
frame_disturbance= frame_disturbance./R;
frame_disturbance_asym= frame_disturbance_asym./R;

%We obtain D_indicator and A_indicator through an aggregation over time
D_indicator= Lpq_weight(frame_disturbance, D_POW_S, D_POW_T);
A_indicator= Lpq_weight(frame_disturbance_asym, A_POW_S, A_POW_T);
average_power_diff = average_power_x - average_power_y;
```

```
abs_average_power_diff = vabs(average_power_diff);
FreqDistance = sum(abs_average_power_diff);
%FreqDistance is the integration over frequency of the difference between
%the two average power.
end
```

Music Parameters: file intensity_warping_of.m

```
function [loudness_dens]=intensity_warping_of(N, centre_bark,
                                                pitch_pow_dens, threshold)
%Loudness density is obtained through the Zwicker's law
ZWICKER_POWER=0.23; S1=1.866055e-1;
  for c=1:N,
      for r=1:49,
          if centre_bark(r)<4,
              h=6/(centre_bark(r)+2);
          else,
              h=1;
          end;
          if h>2, h=2; end;
          h=h^0.15;
          modified_zwicker_power=ZWICKER_POWER*h;
          if pitch_pow_dens(r,c)>threshold(r),
            loudness_dens(r,c)=((threshold(r)/0.5)^modified_zwicker_power)*
  ((0.5+ 0.5*pitch_pow_dens(r,c)/threshold(r))^(modified_zwicker_power)-1);
          else,
              loudness_dens(r,c)=0;
          end;
          loudness_dens(r,c)=loudness_dens(r,c)*S1;
      end;
```

```
  end;
end
```

Music Parameters: file pseudo_Lp3.m

```
function [frame]=pseudo_Lp3(N, width, dist) frame=zeros(1, N);
prod=zeros(49, 1); H=vabs(dist);
  for t=1:N,
      prod=width.*H(:,t);
      DAn= sum(prod)
      frame(t)= DAn;
  end;
  %frame is a vector of N elements
end
```

Music Parameters: file Lpq_weight.m

```
function [result_time]=Lpq_weight(frame_dist, power_syll, power_time)
%Division in syllables
[syllable, zf]=buffer(frame_dist, 20, 10);
Numsyll=size(syllable,2);
res=syllable.^power_syll; %res is a 20xNumsyll matrix
%Aggregation over time
  for c=1:Numsyll,
      result_syllable(c)=sum(res(:,c));
  end;
result_syllable= result_syllable.^(1/power_syll);
res_time= result_syllable.^power_time; %res_time is a Numsyll vector
result_time= (sum(res_time))/Numsyll;
result_time=(result_time)^(1/power_time);
end
```

Video Parameters: file general.m

```
function [si_gain, si_loss, hv_loss, hv_gain]= general

%This program calculates the parameters Sigain, Siloss, Hvgain, Hvloss
%The other two files are subprograms


%Here the tests signals are the original cinema trailer (trailer_orig) and
%the degraded cinema traier encoded with H.263 and AMR. Both signals must
%be in AVI format.
mov_orig =
aviread('C:\MATLAB7\work\VideoParameters\trailer_orig'); orig=
size(mov_orig); Norig= orig(1,2); mov_deg =
aviread('C:\MATLAB7\work\VideoParameters\tr_H263_AMR_75@7,9');
deg= size(mov_deg); Ndeg= deg(1,2);

 if Ndeg<Norig,
   aviInf=aviinfo('C:\MATLAB7\work\VideoParameters\tr_H263_AMR_75@7,9');
   aviInf.ImageType='truecolor';
   F=aviInf.NumFrames; %F is the number of frames in the video.
 else,
   aviInf=aviinfo('C:\MATLAB7\work\VideoParameters\trailer_orig');
   aviInf.ImageType='truecolor';
   F=aviInf.NumFrames; %F is the number of frames in the video.
 end;

 NumReg = fix(F/5); %Fix rounds the number toward zero

 %Through "region" I obtain 4 matrixes that contain the parameters for
%each spatial-temporal region.
```

```
[log_gaintot, ratio_losstot, ratio_loss_matr, log_gain_matr]=
 region(NumReg, mov_orig, mov_deg)

%Now I calculate the Spatial and then Temporal Collapsing Function on
%log_gaintot in order to obtain Sigain parameter
  for t=1:NumReg,
      meansp(1,t)=mean(log_gaintot(1:396,t)); %396 is the number of the
                                              %columns. meansp is a row vector
  end;
 meantp=mean(meansp(1,1:NumReg)); %meantp is a number

  %Now I apply clip_0.004
   if meantp>0,
         si_gain= max(meantp, 0.004)-0.004;
   else,
         si_gain= min(meantp, -0.004)+0.004;
   end;

  %Upper threshold of 0.14
   if si_gain>0.14,
         si_gain= 0.14;
   end;

 %Now I calculate the Spatial and Temporal Collapsing Function on
 %ratio_losstot in order to obtain Siloss parameter

 %Spatial Collapsing function:
  A= ratio_losstot';
    for t=1:NumReg,
        B(t, 1:396)=sort(A(t,1:396)); %It sorts the rows of A from low to
```

```
        %high in a matrix B. It takes only the smallest 20 values
        %(20 because it's 5% of 396)
        Below(t,1)= mean(B(t, 1:20)); %In every row of Below there are the
                                      %mean values of the rows of C
    end;
  %Below is the result of the spatial function. It is a column of NumReg
  %elements

  %Temporal Collapsing Function
  T=sort(Below);
  si_loss= T((ceil(0.1*NumReg)), 1);

%Now I calculate the Spatial and Temporal Collapsing Function on
%ratio_losstot in order to obtain Hvloss parameter

  A = ratio_loss_matr';
  % Spatial Collapsing Function:
    for t=1:NumReg,
        B(t, 1:396)=sort(A(t,1:396));
          below(t,1)= mean(B(t,1:20));
    end;
  %Below is the result of the spatial function. It is a column of NumReg
  %elements

  %Temporal Collapsing Function + nonlinear scaling:
  temporal_mean_square = (mean (below (1:NumReg , 1)))^2;
  hv_loss = (max(temporal_mean_square, 0.06))-0.06;  %clip function

%Now I calculate the Spatial and Temporal Collapsing Function on
%ratio_losstot in order to obtain Hvgain parameter
```

```
   C=log_gain_matr'
  %Spatial Collapsing Function:
    for t=1:NumReg,
     D(t, 1:396)=sort(C(t,1:396)); %It sorts the rows of A from low to high
     above(t,1)= mean(D(t, 377:396)); %In every row of Below there are the
                                      %mean values of the rows o C
    end;

  %Temporal Collapsing function:
  hv_gain = mean (above (1:NumReg , 1));

end
```

Video Parameters: file region.m

```
function [log_gaintot, ratio_losstot, ratio_loss_matr,
log_gain_matr]=
                  region(NumReg, mov_original, mov_degraded)


for x=1:NumReg,
       mov_orig = mov_original(x*5-4:x*5); %each STregion is formed by 5
       %temporal frames. So mov_orig are only 5 frames.
       %'Feature' calculates the features for the 396 regions contained in
       %5 frames
      [fhv13_vett_orig, stdev_orig_intatt]= feature(mov_orig);
       %fhv13_vett_orig and stdev_orig_intatt are vectors of 396 elements
      stdev_origR=stdev_orig_intatt;
      for i= 1:396,
          if stdev_origR(i)<12,
```

```
            stdev_origR(i)=12;
         end;
    end;

mov_deg = mov_degraded(x*5-4:x*5);
[fhv13_vett_deg, stdev_deg_intatt]= feature(mov_deg); %stdev_deg is a
                                                      %vector
 stdev_degR=stdev_deg_intatt;
   for i= 1:396,
      if stdev_degR(i)<12,
      stdev_degR(i)=12;
      end;
   end;

 %ratio_loss function is calculated on stdev (standard deviation)
  for i= 1:396,
      d= (stdev_degR(i)-stdev_origR(i))/stdev_origR(i);
        if d>0,
         ratio_loss(i,1)=0;
        else,
         ratio_loss(i,1)=d;
       end;
  end;

stdev_origL=stdev_orig_intatt;
 for i= 1:396,
   if stdev_origL(i)<8,
      stdev_origL(i)=8;
   end;
 end;
```

```
stdev_degL= stdev_deg_intatt;
  for i= 1:396,
   if stdev_degL(i)<8,
      stdev_degL(i)=8;
   end;
  end;

 %log_gain function is calculated on stdev (standard deviation)
  for i= 1:396,
    log_gain(i,1)=log10(stdev_degL(i)/stdev_origL(i));
      if log_gain(i,1)<0,
        log_gain(i,1)=0;
      end;
  end;

  %Untill here I have ratio_loss and log_gain that are vectors of 396
  %elements. They are calculated every 5 frames.


     if x==1, %If I take the first 5 frames, log_gaintot is a vector of
              %396 elements
         log_gaintot= horzcat(log_gain);
     else,
         log_gaintot= horzcat(log_gaintot,log_gain);
     end;
    % When the parameters are calculated for the whole stream (all
    % frames), log_gaintot is a matrix of 396 rows and NumReg columns.

     if x==1, %If I take the first 5 frames, ratio_losstot is a vector
```

```
            %of 396 elements
        ratio_losstot= horzcat(ratio_loss);
    else,
        ratio_losstot= horzcat(ratio_losstot,ratio_loss);
     end;
  % When the parameters are calculated for the whole stream (all
  % frames), ratio_losstot is a matrix of 396 rows and NumReg columns.

  %At the end of the cycle, log_gaintot and ratio_losstot contain the
  %396 x NumReg  parameters calculated on the standard deviation

  %Now it starts the calculation of the parameters constructed on
  %fhv13 feature.

  %ratio_loss function upon fhv13
  ratio_loss_vett=zeros(396,1);
for i= 1:396,
  ratio_loss_vett(i) = (fhv13_vett_deg(i)-fhv13_vett_orig(i))/
                       fhv13_vett_orig(i);
    if ratio_loss_vett(i) > 0,
        ratio_loss_vett(i) = 0;
    end;
end;

 %log_gain function upon fhv13
 log_gain_vett=zeros(396,1);
for i= 1:396,
  log_gain_vett (i)= log10 (fhv13_vett_deg(i)/fhv13_vett_orig(i));
    if log_gain_vett (i) < 0,
       log_gain_vett (i) =0;
```

```
        end;
    end;



    if x==1,
           ratio_loss_matr = horzcat(ratio_loss_vett);
    else,
           ratio_loss_matr = horzcat(ratio_loss_matr,ratio_loss_vett);
    end;

    if x==1,
           log_gain_matr = horzcat(log_gain_vett);
    else,
           log_gain_matr = horzcat(log_gain_matr,log_gain_vett);
    end;
    % When the parameters are calculated for the whole stream (all
    % frames), log_gain_vett and ratio_loss_matr are matrixes of
    %396 rows and NumReg columns.

end;

end
```

Video Parameters: file feature.m

```
function [fhv13_vett, stdevtot]=feature(mov)

%It calculates the fhv13 and stdev features on the 396 regions (indifferent
%if on degradated or original video) that corrispond to 5 frames
n=144;
m=176; %Resolution in case of QCIF
```

```
%Perceptual filters to enhance edge information:
row=[-.0052625, -.0173446, -.0427401, -.0768961, -.0957739,
-.0696751, 0,
      .0696751, .0957739, .0768961, .0427401, .0173446, .0052625];
hsob=[row; row; row; row; row; row; row; row; row; row; row; row;
row]; vsob= hsob';

%each ST region is formed by 8 x 8 pixel and 5 frames
for r=1:8:137,
    for c=1:8:169,
        for j=1:5,
          frameST=mov(j).cdata;
          y=rgb2ycbcr(frameST); %I take the luminance of only one frame
          Y=y(:,:,1);
          H=filter2(hsob,Y,'same'); %These are the filter responses. They
                                    %have the same size as Y.
          V=filter2(vsob,Y,'same');

            for w=r:r+7,
                for z=c:c+7,
                 STh((w-r+1),(z-c+1))=H(w,z); %I take a subregion of 8x8
                                              %pixel from the filter responses
                 STv((w-r+1),(z-c+1))=V(w,z);
                end;
            end;

          if j==1,
            STregionh= vertcat(STh);
            STregionv= vertcat(STv);
```

```
        else,
          STregionh= vertcat(STregionh, STh);
          STregionv= vertcat(STregionv, STv);
        end;
      end;
      %I have defined the STregions.
      %STregionh ans STregionv are metrixes of 40 rows and 8 columns

  %Now I calculate the standard deviation on the ST region
  R= STregionh.^2+ STregionv.^2;
  R=sqrt(R);
%FIRST FEATURE
  stdev =std(reshape(R,320,1)); %320 is the mass number of pixel

   if ((r==1) & (c==1)),
       stdevtot= vertcat(stdev);
   else,
       stdevtot= vertcat(stdevtot, stdev);
   end;
   %stedvtot is a vector of 396 values at the end of the cycle
%END OF THE FIRST FEATURE

%SECOND FEATURE
  dt=0.225;
  rmin=20;

  for m=1:8,
      for n=1:40,
          if STregionh(n,m)==0,
             teta(n,m)=1.5708;
```

```
            else,
             teta(n,m)=atan(STregionv(n,m)/STregionh(n,m));
            end;
             if ((R(n,m)>= rmin)&((-pi/2<=teta(n,m)&teta(n,m)<-pi/2+dt)|
    (-dt<teta(n,m)&teta(n,m)<+dt)|(pi/2-dt<teta(n,m)&teta(n,m)<=pi/2+dt))),
                 HV(n,m)=R(n,m);
             else,
                 HV(n,m)=0;
             end;
             if ((R(n,m)>= rmin)&((-pi/2+dt<=teta(n,m)&teta(n,m)<=-dt)|
                 (+dt<=teta(n,m)&teta(n,m)<=pi/2-dt))),
                 HVneg(n,m)=R(n,m);
             else,
                 HVneg(n,m)=0;
             end;
        end;
    end;
    p=3;
    meanHV = sum(sum(HV))/320;

    if meanHV < p,
        meanHV=p;
    end;
    meanHVneg = sum(sum(HVneg))/320;
    if meanHVneg < p,
        meanHVneg=p;
    end;
    fhv13=meanHV/meanHVneg;
```

```
      if ((r==1) & (c==1)),
          fhv13_vett = vertcat(fhv13);
      else,
          fhv13_vett = vertcat(fhv13_vett, fhv13);
      end;
      %fhv13_vett is a vector of 396 values at the end of the cycle
    %END OF THE SECOND FEATURE


    end;
end;

end
```

APPENDIX B: Tests results

In this appendix all the tests results and the tests questionnaire are enclosed.

AUDIOVISUAL TESTS		ORIGINAL RESULTS			ADJUSTED RESULTS					
Description	**TestNumber**	**1° round MOSav**	**2° round MOSav**	**TOT MOSav**	**1° round MOSav**	**2° round MOSav**	**TOT MOSav**	bit audio	bit video	bit tot
training_MPEG-4_AAC_105_48.3gp	00A	2,882353	3,352941	3,117647	2,9375	3,3125	3,125	48	57	105
training_H.263_AAC_56_8.3gp	00B	1,529412	1,411765	1,470588	1,529412	1,411765	1,470588	8	48	56
training_H.263_AMR_75_12,2.3gp	00C	3,705882	3,529412	3,617647	3,75	3,4375	3,59375	12,2	62,8	75
tr_H.263_AAC_56_8.3gp	1	2,352941	2,117647	2,235294	2,352941	2,117647	2,235294	8	48	56
vc_MPEG-4_AMR_105_12,2.3gp	2	3,647059	3	3,323529	3,5625	3,0625	3,3125	12,2	92,8	105
vm_H.263_AAC_105_48.3gp	3	2,352941	2,647059	2,5	2,352941	2,647059	2,5	48	57	105
vc_MPEG-4_AAC_75_24.3gp	4	3,411765	3,117647	3,264706	3,3125	3,125	3,21875	24	51	75
tr_MPEG-4_AAC_105_32.3gp	5	4,117647	3,941176	4,029412	4,117647	3,941176	4,029412	32	73	105
vm_MPEG-4_AMR_75_7,9.3gp	6	2,294118	1,764706	2,029412	2,2	1,866667	2,033333	7,9	67,1	75
vc_MPEG-4_AAC_56_8.3gp	7	1,882353	1,352941	1,617647	1,666667	1,333333	1,5	8	48	56
vm_MPEG-4_AAC_75_16.3gp	8	2,882353	3,470588	3,176471	3	3,4	3,2	16	59	75
tr_MPEG-4_AMR_56_10,2.3gp	9	2,588235	2,294118	2,441176	2,5	2,3125	2,40625	10,2	45,8	56
vc_MPEG-4_AAC_56_16.3gp	10	2,352941	1,882353	2,117647	2,266667	2	2,133333	16	40	56
vm_MPEG-4_AMR_56_7,9.3gp	11	1,529412	1,823529	1,676471	1,529412	1,823529	1,676471	7,9	48,1	56
vc_H.263_AAC_105_32.3gp	12	3,588235	3,764706	3,676471	3,588235	3,764706	3,676471	32	73	105
tr_H.263_AAC_105_48.3gp	13	2,823529	3	2,911765	2,75	3,0625	2,90625	48	57	105
vc_MPEG-4_AMR_105_5,9.3gp	14	3,058824	2,411765	2,735294	2,9375	2,375	2,65625	5,9	99,1	105
vm_H.263_AMR_75_5,9.3gp	15	1,352941	1,235294	1,294118	1,352941	1,235294	1,294118	5,9	69,1	75
tr_MPEG-4_AMR_56_5,9.3gp	16	1,588235	1,529412	1,558824	1,5	1,5625	1,53125	5,9	50,1	56
vc_MPEG-4_AMR_75_5,9.3gp	17	3,058824	2,882353	2,970588	2,9375	2,9375	2,9375	5,9	69,1	75
tr_MPEG-4_AAC_75_16.3gp	18	3,882353	4,058824	3,970588	3,666667	3,833333	3,75	16	59	75
vm_H.263_AMR_105_7,9.3gp	19	2,470588	2,294118	2,382353	2,470588	2,294118	2,382353	7,9	97,1	105
vc_H.263_AAC_75_16.3gp	20	2,647059	2,705882	2,676471	2,647059	2,705882	2,676471	16	59	75
tr_H.263_AMR_56_5,9.3gp	21	1,588235	1,647059	1,617647	1,588235	1,647059	1,617647	5,9	50,1	56
vc_H.263_AMR_75_5,9.3gp	22	3	2,764706	2,882353	3	2,764706	2,882353	5,9	69,1	75
vm_MPEG-4_AMR_56_10,2.3gp	23	2,235294	2,176471	2,205882	2,235294	2,176471	2,205882	10,2	45,8	56
vc_H.263_AMR_56_7,9.3gp	24	2,470588	2,529412	2,5	2,470588	2,529412	2,5	7,9	48,1	56
tr_H.263_AAC_75_16.3gp	25	2,764706	3,176471	2,970588	2,928571	3	2,964286	16	59	75
vc_H.263_AMR_105_12,2.3gp	26	3,411765	3,588235	3,5	3,375	3,6875	3,53125	12,2	92,8	105
vm_H.263_AMR_56_7,9.3gp	27	1,647059	1,823529	1,735294	1,647059	1,823529	1,735294	7,9	48,1	56
vc_H.263_AAC_105_24.3gp	28	3,117647	3,294118	3,205882	3,117647	3,294118	3,205882	24	81	105
tr_H.263_AAC_56_16.3gp	29	1,823529	2	1,911765	1,875	1,9375	1,90625	16	40	56
vm_MPEG-4_AAC_56_8.3gp	30	1,235294	1,235294	1,235294	1,235294	1,235294	1,235294	8	48	56
tr_MPEG-4_AAC_75_32.3gp	31	3,176471	3,823529	3,5	3,285714	3,642857	3,464286	32	43	75
vm_MPEG-4_AAC_105_32.3gp	32	4,176471	4,294118	4,235294	4,25	4,25	4,25	32	73	105
vc_H.263_AMR_105_7,9.3gp	33	2,941176	3,176471	3,058824	2,875	3,25	3,0625	7,9	97,1	105
tr_MPEG-4_AAC_105_24.3gp	34	4,117647	4,294118	4,205882	4,133333	4,333333	4,233333	24	81	105
vc_MPEG-4_AAC_75_16.3gp	35	2,705882	2,882353	2,794118	2,666667	2,866667	2,766667	16	59	75
vm_H.263_AAC_105_24.3gp	36	2,529412	3	2,764706	2,5625	2,9375	2,75	24	81	105
tr_H.263_AMR_75_7,9.3gp	37	1,705882	1,764706	1,735294	1,705882	1,764706	1,735294	7,9	67,1	75
vc_H.263_AMR_75_12,2.3gp	38	2,941176	2,705882	2,823529	2,733333	2,8	2,766667	12,2	62,8	75
vm_MPEG-4_AAC_56_16.3gp	39	1,705882	1,647059	1,676471	1,705882	1,647059	1,676471	16	40	56
vc_H.263_AMR_105_5,9.3gp	40	2,941176	2,823529	2,882353	2,714286	2,785714	2,75	5,9	99,1	105
tr_MPEG-4_AMR_75_7,9.3gp	41	2	2,352941	2,176471	2	2,133333	2,066667	7,9	67,1	75
vc_MPEG-4_AAC_105_24.3gp	42	3,176471	3,705882	3,441176	3,25	3,6875	3,46875	24	81	105
vm_H.263_AMR_105_12,2.3gp	43	2,411765	2,705882	2,558824	2,411765	2,705882	2,558824	12,2	92,8	105
tr_H.263_AMR_105_12,2.3gp	44	2,764706	2,705882	2,735294	2,733333	2,666667	2,7	12,2	92,8	105
vm_MPEG-4_AMR_75_5,9.3gp	45	1,647059	1,764706	1,705882	1,6875	1,6875	1,6875	5,9	69,1	75
vc_MPEG-4_AMR_56_10,2.3gp	46	2,529412	2,705882	2,617647	2,4375	2,75	2,59375	10,2	45,8	56
tr_H.263_AMR_105_5,9.3gp	47	1,764706	1,647059	1,705882	1,625	1,625	1,625	5,9	99,1	105
vm_MPEG-4_AAC_75_32.3gp	48	2,823529	3,294118	3,058824	2,9375	3,3125	3,125	32	43	75
tr_MPEG-4_AMR_105_5,9.3gp	49	2	1,882353	1,941176	2	1,882353	1,941176	5,9	99,1	105
vm_H.263_AAC_75_16.3gp	50	2,470588	2,882353	2,676471	2,5	2,8125	2,65625	16	59	75

Figure 7.14: Mean values on the results of the audiovisual tests pro sequence.

AUDIOVISUAL TESTS		ORIGINAL RESULTS			ADJUSTED RESULTS					
Description	TestNumber	1° round MOSav	2° round MOSav	TOT MOSav	1° round MOSav	2° round MOSav	TOT MOSav	bit audio	bit video	bit tot
tr_H.263_AAC_56_24.3gp	51	1,647059	2,235294	1,941176	1,6875	2,1875	1,9375	24	32	56
vm_H.263_AAC_56_16.3gp	52	1,411765	1,352941	1,382353	1,411765	1,352941	1,382353	16	40	56
tr_H.263_AMR_56_7,9.3gp	53	1,470588	1,705882	1,588235	1,470588	1,705882	1,588235	7,9	48,1	56
vc_H.263_AMR_56_5,9.3gp	54	2,705882	2,705882	2,705882	2,5625	2,6875	2,625	5,9	50,1	56
vm_MPEG-4_AAC_75_24.3gp	55	3,705882	4,176471	3,941176	3,705882	4,176471	3,941176	24	51	75
vc_MPEG-4_AMR_75_12,2.3gp	56	2,9375	3,058824	2,998162	2,9375	3,0625	3	12,2	62,8	75
tr_H.263_AMR_56_10,2.3gp	57	2,294118	2,352941	2,323529	2,294118	2,352941	2,323529	10,2	45,8	56
vm_MPEG-4_AMR_105_12,2.3gp	58	2,882353	2,823529	2,852941	2,866667	2,8	2,833333	12,2	92,8	105
vc_MPEG-4_AMR_56_5,9.3gp	59	2,470588	2,647059	2,558824	2,470588	2,647059	2,558824	5,9	50,1	56
vm_H.263_AAC_75_24.3gp	60	2,470588	2,941176	2,705882	2,5	2,875	2,6875	24	51	75
tr_MPEG-4_AAC_75_24.3gp	61	3,470588	3,588235	3,529412	3,470588	3,588235	3,529412	24	51	75
vm_H.263_AAC_56_8.3gp	62	1,176471	1,294118	1,235294	1,1875	1,1875	1,1875	8	48	56
tr_H.263_AAC_75_24.3gp	63	2,882353	2,875	2,878676	2,9375	2,875	2,90625	24	51	75
vm_MPEG-4_AAC_56_24.3gp	64	2,764706	2,941176	2,852941	2,764706	2,941176	2,852941	24	32	56
tr_MPEG-4_AAC_105_48.3gp	65	3,647059	3,529412	3,588235	3,733333	3,6	3,666667	48	57	105
vm_H.263_AMR_75_12,2.3gp	66	1,941176	1,882353	1,911765	1,941176	1,882353	1,911765	12,2	62,8	75
tr_MPEG-4_AMR_56_7,9.3gp	67	1,764706	1,764706	1,764706	1,764706	1,764706	1,764706	7,9	48,1	56
vc_H.263_AMR_75_7,9.3gp	68	2,941176	2,941176	2,941176	2,875	3	2,9375	7,9	67,1	75
vm_MPEG-4_AMR_105_5,9.3gp	69	1,941176	1,647059	1,794118	1,8125	1,625	1,71875	5,9	99,1	105
vc_MPEG-4_AMR_75_7,9.3gp	70	2,882353	2,941176	2,911765	2,882353	2,941176	2,911765	7,9	67,1	75
tr_H.263_AAC_75_32.3gp	71	2,411765	2,764706	2,588235	2,4375	2,6875	2,5625	32	43	75
vm_MPEG-4_AMR_105_7,9.3gp	72	2,352941	2	2,176471	2,352941	2	2,176471	7,9	97,1	105
vc_MPEG-4_AAC_105_32.3gp	73	3,705882	4	3,852941	3,928571	4,071429	4	32	73	105
tr_MPEG-4_AMR_105_7,9.3gp	74	2,235294	2,235294	2,235294	2,4	2,133333	2,266667	7,9	97,1	105
vm_MPEG-4_AAC_105_24.3gp	75	4,117647	4,470588	4,294118	4,266667	4,4	4,333333	24	81	105
tr_MPEG-4_AAC_56_16.3gp	76	2,058824	2,294118	2,176471	2,153846	2,153846	2,153846	16	40	56
vc_MPEG-4_AMR_56_7,9.3gp	77	2	2,294118	2,147059	2,0625	2,25	2,15625	7,9	48,1	56
vm_H.263_AAC_105_32.3gp	78	2,529412	3	2,764706	2,833333	3	2,916667	32	73	105
tr_H.263_AAC_105_24.3gp	79	3,411765	3,588235	3,5	3,411765	3,588235	3,5	24	81	105
vc_H.263_AAC_56_16.3gp	80	1,647059	1,705882	1,676471	1,647059	1,705882	1,676471	16	40	56
vm_MPEG-4_AMR_56_5,9.3gp	81	1,176471	1,411765	1,294118	1,176471	1,411765	1,294118	5,9	50,1	56
tr_MPEG-4_AMR_75_12,2.3gp	82	2,470588	2,470588	2,470588	2,4375	2,5625	2,5	12,2	62,8	75
vm_H.263_AMR_75_7,9.3gp	83	1,529412	1,588235	1,558824	1,529412	1,588235	1,558824	7,9	67,1	75
vc_H.263_AAC_75_24.3gp	84	2,823529	2,764706	2,794118	2,823529	2,764706	2,794118	24	51	75
tr_MPEG-4_AMR_75_5,9.3gp	85	2	1,588235	1,794118	1,875	1,625	1,75	5,9	69,1	75
vm_H.263_AAC_56_24.3gp	86	2	2,352941	2,176471	2	2,133333	2,066667	24	32	56
tr_MPEG-4_AAC_56_24.3gp	87	2,764706	2,882353	2,823529	2,764706	2,882353	2,823529	24	32	56
vc_H.263_AAC_56_8.3gp	88	1,588235	1,529412	1,558824	1,588235	1,529412	1,558824	8	48	56
vm_MPEG-4_AAC_105_48.3gp	89	3,588235	3,647059	3,617647	3,5625	3,75	3,65625	48	57	105
vc_H.263_AMR_56_10,2.3gp	90	2,764706	2,705882	2,735294	2,764706	2,705882	2,736294	10,2	45,8	56
tr_MPEG-4_AAC_56_8.3gp	91	2,176471	2,117647	2,147059	2,125	2,1875	2,15625	8	48	56
vc_MPEG-4_AMR_105_7,9.3gp	92	2,411765	2,294118	2,352941	2,411765	2,294118	2,352941	7,9	97,1	105
vm_H.263_AMR_56_10,2.3gp	93	1,764706	1,705882	1,735294	1,764706	1,705882	1,735294	10,2	45,8	56
tr_MPEG-4_AMR_105_12,2.3gp	94	2,764706	2,470588	2,617647	2,6875	2,5	2,59375	12,2	92,8	105
vm_MPEG-4_AMR_75_12,2.3gp	95	2,470588	2,823529	2,647059	2,5	2,75	2,625	12,2	62,8	75
tr_H.263_AAC_105_32.3gp	96	3,411765	3,470588	3,441176	3,411765	3,470588	3,441176	32	73	105
vm_H.263_AAC_75_32.3gp	97	2,882353	2,823529	2,852941	2,8125	2,875	2,84375	32	43	75
tr_H.263_AMR_75_5,9.3gp	98	1,470588	1,529412	1,5	1,470588	1,529412	1,5	5,9	69,1	75
vm_H.263_AMR_105_5,9.3gp	99	1,470588	1,588235	1,529412	1,5	1,5	1,5	5,9	99,1	105
tr_H.263_AMR_75_12,2.3gp	100	2,411765	2,235294	2,323529	2,266667	2,333333	2,3	12,2	62,8	75
vm_H.263_AMR_56_5,9.3gp	101	1,411765	1,529412	1,470588	1,411765	1,529412	1,470588	5,9	50,1	56
tr_H.263_AMR_105_7,9.3gp	102	2,176471	2	2,088235	2,125	2,0625	2,09375	7,9	97,1	105

Figure 7.15: Mean values on the results of the audiovisual tests pro sequence.

AUDIO TESTS		ORIGINAL RESULTS			ADJUSTED RESULTS			
Description	TestNumber	1° round MOSa	2° round MOSa	TOT MOSa	1° round MOSa	2° round MOSa	TOT MOSa	bit audio
training_MPEG-4_AAC_105_48.3gp	A	3,2	4	3,6	3,777778	3,888889	3,833333	48
training_H.263_AAC_56_8.3gp	B	2,2	2,4	2,3	2,555556	2,333333	2,444444	8
training_H.263_AMR_75_12,2.3gp	C	3,4	3,7	3,55	3,666667	3,888889	3,777778	12,2
tr_H.263_AAC_56_8.3gp	1	1,8	2,2	2	2	2,2	2,1	8
vm_H.263_AAC_105_48.3gp	2	4,5	4,2	4,35	4,5	4,2	4,35	48
tr_H.263_AAC_105_48.3gp	3	3	3	3	3,125	3,125	3,125	48
vm_H.263_AMR_75_12,2.3gp	4	1,9	2,4	2,15	1,9	2,4	2,15	12,2
vc_H.263_AMR_56_7,9.3gp	5	3,4	3,4	3,4	3,222222	3,444444	3,333333	7,9
vm_H.263_AMR_56_7,9.3gp	6	2,1	2,3	2,2	2,1	2,3	2,2	7,9
tr_H.263_AAC_56_16.3gp	7	2,2	2,4	2,3	2,2	2,4	2,3	16
vc_H.263_AMR_75_12,2.3gp	8	3,5	3,7	3,6	3,5	3,7	3,6	12,2
tr_H.263_AAC_56_24.3gp	9	3,7	3,7	3,7	3,555556	3,777778	3,666667	24
vm_H.263_AAC_56_16.3gp	10	2,6	2,5	2,55	2,6	2,5	2,55	16
tr_H.263_AMR_56_7,9.3gp	11	1,2	1,2	1,2	1	1,222222	1,111111	7,9
vm_H.263_AAC_75_32.3gp	12	4,4	4,3	4,35	4,4	4,3	4,35	32
tr_H.263_AMR_56_10,2.3gp	13	1,7	1,3	1,5	1,444444	1,222222	1,333333	10,2
vm_H.263_AAC_56_8.3gp	14	1,6	1,6	1,6	1,6	1,6	1,6	8
vc_H.263_AMR_56_10,2.3gp	15	3,8	3,5	3,65	3,875	3,5	3,6875	10,2
tr_H.263_AAC_75_32.3gp	16	3,7	3,7	3,7	3,7	3,7	3,7	32
vc_H.263_AAC_56_16.3gp	17	1,9	2,3	2,1	1,9	2,3	2,1	16
vm_H.263_AAC_56_24.3gp	18	3,9	3,8	3,85	3,9	3,8	3,85	24
vc_H.263_AAC_56_8.3gp	19	1,6	1,8	1,7	1,6	1,8	1,7	8
tr_H.263_AMR_56_5,9.3gp	20	1	1	1	1	1	1	5,9
vm_H.263_AMR_56_10,2.3gp	21	2,3	2	2,15	2,111111	2	2,055556	10,2
vc_H.263_AMR_56_5,9.3gp	22	3,5	3,5	3,5	3,5	3,5	3,5	5,9
tr_H.263_AMR_75_12,2.3gp	23	1,9	2	1,95	1,9	2	1,95	12,2
vm_H.263_AMR_56_5,9.3gp	24	1,7	1,4	1,55	1,7	1,4	1,55	5,9
vc_MPEG-4_AAC_24	25	4,25	4,75	4,5	4,25	4,75	4,5	24
vc_MPEG-4_AAC_32	26	4	4	4	4	4	4	32

Figure 7.16: Mean values on the results of the audio tests pro sequence.

Test Number: Round: Date:

Name:

Age: **Sex**: Female / Male

Education: primary school secondary school university

Occupation: student other:

Which mobile phone do you have? brand: model:

Have you some experience in image processing (films, photos, etc)? Yes / No

How often do you send SMS?
daily weekly sometimes never

How often do you send MMS?
daily weekly sometimes never not supported by phone

How much do you pay monthly for your mobile phone bill?
0,- up to 20,-€ 20, - up to 40,-€ 40, - up to 70,-€
70, - up to 100,-€ beyond 100 €

Total Quality

Clip number	**Evaluation**				
1	1	2	3	4	5
	Bad	Poor	Fair	Good	Excellent

Assume that you have paid for this clip. Would you be satisfied with the quality of the video clip?
Yes No

2	1	2	3	4	5
	Bad	Poor	Fair	Good	Excellent

Assume that you have paid for this clip. Would you be satisfied with the quality of the video clip?
Yes No

Figure 7.17: Model of our test questionnaire.

Bibliography

[1] S. Winkler. *Digital Video Quality. Vision Models and Metrics.* Genista Corporation, Montreux, Switzerland, Jan. 2005, Pages: 38-39.

[2] 3GPP TS 26.234 (release 4). *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs.* Dec. 2002.

[3] ITU-T Recommendation H.263. *AnnexX: Profiles and levels definition.* Apr. 2001.

[4] F. G. Lönnberg. *Adaptive Multi Rate Codec. Cost effective route to network capacity expansion.* Copyright 2003 Nokia, Nov. 2003

[5] 3Gpp TS 26.071 (release 4). *3rd Generation Partnership Project; Technical Specification Group Services and Sytem Aspects; Mandatory Speech Codec speech processing functions; AMR Speech Codec; General Description.* March 2001.

[6] 3Gpp TS 26.090. *Transcoding functions.* Apr. 2001.

[7] 3GPP TS 26.403 V 2.0.0 (Release 6). *Enhanced aacPlus audio code; Encoder specification AAC part*, Sep. 2004.

[8] ISO/IEC 14496-3:2001, *Information technology - Coding of audio-visual objects - Part 3: Audio*, 2001

[9] ISO/IEC 14496-3:2001/ Amd.2:2004, *Parametric Coding for High Quality Audio*, 2001

[10] S. Voran. *Objective Estimation of Perceived Speech Quality. Part II: Evaluation of the Measuring Normalizing Block Technique.* IEEE transactions on speech and audio processing, vol. 7, no. 4, pp. 385-390, July 1999.

[11] S. Voran. *Objective Estimation of Perceived Speech Quality. Part I: Development of the Measuring Normalizing Block Technique.* IEEE transactions on speech and audio processing, vol. 7, no. 4, July 1999.

[12] ITU-T Recommendation P.862. *Perceptual evaluation of speech quality (PESQ): An objective method for end-to-end speech quality assessment of narrow band telephone networks and speech codecs.* Feb. 2001.

[13] E. Zwicker, R. Feldtkeller. *Das Ohr als Nachrichtenempfänger.* S. Hirzel Verlag, Stuttgart, 1967.

[14] A. Rimell, M. Hollier. *The significance of cross-modal interaction in audio-visual quality perception.* Advanced Communications Research, BT Laboratories, Ipswich, IP5 3RE, UK, 1999.

[15] S. Tasaka, Y. Ishibashi. *Mutually Compensatory Property of Multimedia QoS.* Nagoya Institute of Technology, Nagoya, Japan, IEEE Transactions, 2002.

[16] American National Standard for Telecommunications. T1.801.03-2003. Working Group on Performance, Reliability, and Signal Processing. *Digital Transport of One-Way Video Signals- Parameters for Objective Performance Assessment.* Copyright 2004 by Alliance for Telecommunication Industy Solutions

[17] M. Hol1ier, R. Voelcker. *Towards a multimodal perceptual model.* BT Technology Journal, vol. 15, no. 4, pp. 163-172, 1997.

[18] ITU-T Recommendation P.910 - 1999, *Subjective video quality assessment methods for multimedia application.* International Telecommunication Union.

[19] O. Nemethova , M. Ries, E. Siffel, M. Rupp. *Quality Assessment for H.264 Coded Low-Rate and low-Resolution Video Sequences.* Proc. of Conference on Internet and Information Technologies (CIIT), St. Thomas, US Virgin Islands, Nov. 2004, Pages: 136-140.

[20] O. Nemethova, M. Ries, E. Siffel, M. Rupp. *Subjective Evaluation of Video Quality for H.264 Encoded Sequences.* Institute for Communications and RF Engineering, Wien, Austria, 2004

[21] S. Winkler, F. Dufaux, *Video Quality Evaluation for Mobile Applications*, Audiovisual Communications Laboratory and Signal Processing Laboratory, Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland.

[22] G. Hauske, T. Stockhammer, R. Hofmaier, *Subjective Image Quality of Low-rate and Low-Resolution Video Sequences*, Proc. International Workshop on Mobile Multimedia Communication, Munich, Germany, Oct. 2003.

[23] A. A. Webster, C. T. Jones, M. H. Pinson, S. D. Voran, S. Wolf, *An objective video quality assessment system based on human perception*, Proc. SPIE Human Vision, Processing and digital display IV, vol. 1913, pp. 15-26, San Jose, CA, February 1993.