DISSERTAÇÃO

Mestrado em Engenharia Electrotécnica

3DVQM: 3D Video Quality Monitor

Bruno Tomé dos Santos Feitor

Leiria, novembro de 2013

MASTER DISSERTATION

Electrical Engineering

3DVQM: 3D Video Quality Monitor

Bruno Tomé dos Santos Feitor

Master dissertation performed under the guidance of Professor Pedro Amado António Assunção of Escola Superior de Tecnologia e Gestão of Instituto Politécnico de Leiria.

Leiria, November 2013

Faith it does not makes things
easy, it makes them possible.

(Luke 1:37)

Acknowledgments

I would like to thank to everyone that helped during this research work, and that made it possible to accomplish.

I would like to express my gratitude to my adviser Professor Pedro António Amado Assunção, who launched me in 3DVQM project. I'm thankful for his guidance and availability that were essential for the proper conduct of this research work. I also would like to thank to the research group of 3DVQM project, Professor Luis Cruz and Professor Rui Marinheiro, for sharing their knowledge and helping this research work.

I would like to thank the opportunity of working as researcher in this project that had an important role in my professional an personal progress, and to thank Instituto de Telecomunicações and Escola Superior de Tecnologia e Gestão of IPL, for the laboratory facilities, that gave me conditions to accomplish this work.

I would like to acknowledge my research colleagues of Instituto de Telecomunicações - Leiria, João Carreira, Luís Pinto, Pedro Correia, Luís Lucas and David Ferreira for the friendship and the wonderful work environment. I also want to express my gratitude for my graduation and non-graduation friends, for the moments outside the working environment, friendship and support, which were important along this research work.

The last but not least, a special thanks to my parents, Paulo and Maria, and brothers Samuel, Pedro and sister Daniela to whom I owe all that I have become. Thank you for being there, without you all my accomplishments would not be possible.

Abstract

This dissertation presents a research study and software implementation of an objective quality monitor for 3D video streams transmitted over networks with non-guaranteed packet delivery due to errors, congestion, excessive delay, etc. A review of Video Quality Assessment (VQA) models available in the literature is first presented, addressing 2D and 3D video quality models that were selected as relevant for this research work.

A packet-layer VQA model is proposed based on header information from three different packet-layer levels: Network Abstraction Layer (NAL), Packetised Elementary Streams (PES) and MPEG2 - Transport Stream (TS). Transmission errors leading to undecodable TS packets are assumed to result in a whole frame loss. The proposed method estimates the size of the lost frames, which is used as a model parameter to predict their objective quality, measured as the Structural Similarity Index Metric (SSIM).

In order to materialise the proposed VQA model, a software application was developed that allows monitoring a corrupted 3D video stream quality. To make the monitoring process as user friendly as possible, a Guide User Interface (GUI) was developed. With this feature the user can interact with the application by controlling the input parameters and customizing the results on the output display.

The results show that SSIM of isolated missing stereoscopic frames in 3D coded video can be predicted with Root Mean Square Error (RMSE) accuracy of about 0.1 and Pearson correlation coefficient of 0.8, taking the SSIM of uncorrupted frames as reference. It is concluded that the proposed model is capable of estimating the SSIM quite accurately using only the estimated sizes of single lost frames.

Keywords: 3D video, video quality assessment, monitor's software.

Resumo

Esta dissertação apresenta um trabalho de investigação e implementação do software de um monitor de qualidade objectiva que estima a degradação da qualidade de imagens perdidas em fluxos de vídeo 3D transmitidos através de redes que não garantem a entrega de pacotes devido a erros de transmissão, congestionamento, atrasos, etc. Com o objectivo de resumir os vários modelos de avaliação da qualidade de vídeo (VQA) presentes na literatura, foi feita uma revisão do estado da arte de modelos VQA para conteúdos 2D e 3D que pudessem contribuir para este trabalho de investigação.

O modelo VQA *packet-layer* proposto utiliza apenas informação presente nos cabeçalhos de pacotes de três diferentes camadas: *Network Abstraction Layer* (NAL), *Packetised Elementary Streams* (PES) e *MPEG2 - Transport Stream* (TS). Erros de transmissão que originam pacotes TS danificados correspondem a perda total de imagens. O método proposto estima o tamanho das imagens perdidas, sendo este tamanho usado para determinar a qualidade objectiva da imagem medida em *Structural Similarity Index Metric* (SSIM).

Com o objectivo de materializar o modelo proposto, desenvolveu-se uma aplicação que permite monitorizar a qualidade de um fluxo de vídeo 3D corrompido. Esta aplicação possui uma *Guide User Interface* (GUI) que aproxima o utilizador do processo de monitorização. Esta funcionalidade facilita a interacção do utilizador com o *software* do monitor permitindo controlar os parâmetros de entrada da aplicação, bem como customizar os resultados apresentados na interface.

Os resultados concluem que o SSIM das imagens 3D perdidas pode ser estimado recorrendo ao *Root Mean Square Error* (RMSE) com uma precisão de 0.1 aproximadamente, e recorrendo ao coeficiente de correlação de Pearson de 0.8, utilizando como referência o SSIM das imagens recebidas correctamente. Conclui-se que o modelo proposto permite estimar o SSIM com precisão, utilizando apenas o tamanho das imagens perdidas.

Palavras chave: Vídeo 3D, avaliação da qualidade de vídeo, monitor software.

Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two-Dimensional, p. 7
3D	Three-Dimensional, p. 1
AF	Adaptation Field, p. 24
BLF	Burst Loss Frequency , p. 9
CU	coding units, p. 20
DIBR	depth image based rendering, p. 16
DVB	Digital Video Broadcasting, p. 1
EC	Error Concealment, p. 10
FR	Full Reference, p. 1
GGoP	Group of Group-of-Pictures, p. 14
GOP	Group of Pictures, p. 10
GUI	Guide User Interface, p. v, 39
HVS	Human Visual System, p. 13
HVS	humam visual system, p. 6
IFR	Invalid Frame Rate, p. 10
IP	Internet Protocol, p. 1
ITU	International Telecommunications Union , p. 6
MSE	Mean Square Error, p. 10
MSE	Mean Squared Error, p. 5
MVC	Multiview Video Coding, p. 13

NR	No-Reference, p. 2
OOP	Objective-Oriented Programming, p. 40
PLF	Packet Loss Frequency, p. 11
PLR	Packet Loss Rate, p. 6, 9
PSNR	Peak signal-to-noise ratio, p. 5
QoE	Quality of Experience, p. 1
QoS	Quality of Service, p. 5
RR	Reduced Reference, p. 2
SSIM	Structural Similarity, p. 5, 20
VQA	Video Quality Assessment, p. v, 3–5, 7
VQEG	Video Quality Experts Group, p. 6
VQM	Video Quality Metric, p. 15
fps	frames per second, p. 12

Chapter 1

Introduction

This chapter presents an introduction to the research study carried out in the scope of this dissertation. The motivation and the objectives of the research are highlighted and the structure of the thesis is presented.

1.1 Context and motivation

In recent years, three-dimensional (3D) video quality evaluation has become an increasingly relevant research field, especially when transmission over error prone network is used, such as the near future three-dimensional broadcast services over internet protocol (IP) or digital video broadcasting (DVB) networks [1] and [2]. Considering the users as the common end consumers of multimedia content, they are also the most reliable evaluators of the real quality experienced from the services provided by telecommunications operators. Thus a reliable video quality assessment would surely be based on subjective tests [3] compliant with standardized procedures, that are defined to achieve steady and reliable video quality evaluation through participants [4]). On the one hand, since the human viewers are the final receivers, this method to evaluate the video quality can yield the perfect an desired measures. On the other hand, subjective tests are time consuming due to preparation, requirements and costs. Therefore, collecting information regarding the users quality of experience (QoE) is not a feasible option in most real-time services and applications, particularly in a broadcast scenario. Thus, estimation of subjective quality in coded video streams transmitted over error non-guaranteed channels, using objective metrics (computational algorithms) and simple parameters captured from the coded stream itself and the transmission network, should be developed for practical systems.

Based upon the type of information available for coded video quality evaluation, objective quality methods can be classified into: Full-Reference (FR) , Reduced-Reference (RR)

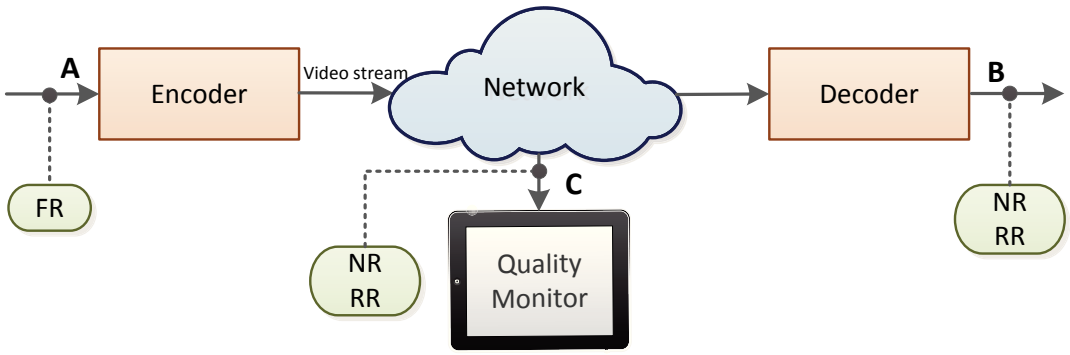


Figure 1.1: Objective video quality measurements.

and No-Reference (NR) methods. FR methods such as those described in [5–7] determine the impaired video quality by comparing the distorted signal with the original one, i.e. the reference. The requirement to access the original video makes the FR methods impractical for video quality monitoring at remote locations from the video encoder, where the reference signal is not readily available. RR methods compute objective quality scores using some kind of reduced information concerning the reference, which is sent through a side channel [8–10]. However, RR methods have similar problems as the FR methods, since video quality assessment needs additional information from the video encoder source. The growing need to measure the quality of compressed video streams at any point of communications channel without needing supplementary information has been fostering increased research attempt on the development of NR quality methods [11–14].

Figure 1.1 depicts several measurement points that can be used to evaluate de objective video quality. Measurement A matches to the uncompressed video available at the encoder's input in the communication system. Measurement B corresponds to the decompressed video available at the output of the decoder. Measurement B corresponds to the encrypted bitstream itself, transmitted along the network. The closer to the decoder that measurement C is taken, the more accurately it can characterize the video that will be displayed to the viewer, because in a error prone network, video information is lost along the network due to packet errors. FR methods compare the distorted signal with the original signal, thus it requires measurements at both A (uncompressed video) and B (decompressed video) stages. RR methods uses some kind of reduced information concerning the reference and thus need measurements at A and either B or C. The original video signal is not required in NR methods, and thus they can predict objective video quality using only measurements at either B or C stages.

This dissertation presents a research study and software implementation of an objective quality monitor for 3D video stream transmitted over error prone networks, in the presence of single frame losses without needing to decode the compressed stream. Figure 1.2 shows

a potential application scenario for the proposed NR 3D video quality monitoring system. Since it only requires information that can be obtained from the transmission network, such as the packets losses and the stream itself, this system is able to operate at any point of the communication path.

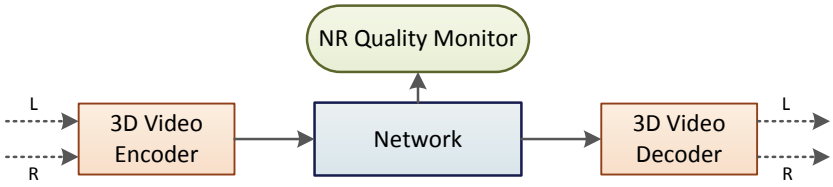


Figure 1.2: NR packet-layer application scenario.

1.2 Objectives

In this context, this research deals with 3D video quality monitoring, with the following objectives.

- **Study the impact of the frame-loss in the perceived 3D video quality.** To determine the perceived video quality in real-time applications NR, VQA models must be used. Considering that video content may be encrypted along the network, decoding the compressed streams is not possible. Objective metrics must be studied from packet layer parameters, which can correlate with the perceived degradation an thus the video quality.

- **Develop a NR Video Quality Assessment (VQA) model for stereoscopic video content.** After finding out the impact of frame-loss in the perceived video quality, the next step is to understand how this correlates and which parameters are important. With such parameters a model must developed to determine de NR objective quality for stereoscopic video.

- **Implement the proposed model in a monitor software application.** Another objective of this dissertation is to develop a monitor application to implement the quality model in a pratical scenario. This application must read the network information that will allow to determine the objective video quality, and must produce real-time graphical information. For that purpose a user interface should be used to make the interaction with the application an intuitive and user-friendly operation.

1.3 Outline

This thesis contains five chapters and two appendixes. The current chapter introduces the research work by presenting its context, motivations and objectives. The following chapters are organised as follows.

Chapter 2 reviews the state-of-the-art concepts related to Video Quality Assessment (VQA) methods for both 2D and 3D video content. The literature review describes several types of VQA methods depending on their requirements and input level information. These methods are analysed to find out which one fulfills this work's goals and specifications. Then several quality models for 2D and 3D video content are presented, to understand how they work and which contribution they might bring to this study.

Chapter 3 presents the proposed NR VQA model for 3D video content, as well as its implementation scenario and results. It is explained how the mathematical model was inferred from the simulation studies, as well as how it was validated for 3D video sequences. The simulation results are discussed in the end of the Chapter.

Chapter 5 concludes this dissertation and presents some suggestions for future work.

Appendix A presents the published papers, which includes some results and conclusions presented in this dissertation, and Appendix B presents the monitor's software documentation.

Chapter 2

Video quality assessment models

Due to the exponential growth of video stream based services over IP networks, it is importante to measure the impact of packet-loss and transmission errors in the perceived video quality. There are two primary methods to evaluate the quality of 3D video: subjective quality assessment and objective quality assessment [15]. The subjective quality assessment method is based in psychological analysis that uses structured experimental designs and human participants to evaluate the quality of the video presented [16]. Accomplishing subjective quality assessment test is a long and expensive process [17]. Besides the need for considering these environmental factors, such as viewing distance, lighting condition, and the choice of these test sequences, more complex issues are also needed to be considered. For example, the observers' conditions [16]. Therefore the interest in no-reference (NR) Video Quality Assessment (VQA) models is increasing, since they measure the objective quality at any stage of the network without needing the reference signal. Due to the increasing amount of 3D video content in broadcast industries, the need for monitoring the Quality of Service (QoS) provided to the clients is gaining a strong momentum. In this context, the NR VQA models are useful to telecommunications operating companies because they allow automatic, standing and real-time monitoring of theirs costumer's Quality of Service (QoS).

A review of objective quality metrics such as Mean Squared Error (MSE) , Peak signal-to-noise ratio (PSNR) and Structural Similarity (SSIM), was made in [18]. It is well known that the most used quality metrics are MSE and PSNR. PSNR is very often used to evaluate the performance of video compression standards (eg. H.264/AVC). The PSNR and MSE mathematical expressions are described next,

$$MSE = \frac{1}{N}\sum_{N}^{n=1}(X_n - Y_n)^2 \tag{2.1}$$

$$PSNR = 10 \log_{10} \frac{L^2}{MSE} \tag{2.2}$$

where X_n and Y_n are the corresponding pixel values in the original and distorted frames, respectively. N is the total pixel numbers in a frame. In PSNR equation, L stands for the maximum possible pixel value of the frame. For example, it will be 255 for the pixels represented by using 8 bits per sample. The two pixel-based methods are easy to computer. Moreover these metrics do not consider the perception of human visual sense. So these evaluating results have a big gap in comparison with the subjective evaluation results. Similar, limitation will occur in 3D video and thus these methods are not that accurate in 3D video.

A popular image/video quality assessment metric that uses structural information was proposed in [19]. It extracts the structural information that is relevant for the quality perceived by the human visual system (HVS) .

$$SSIM(x, y) = \frac{(2u_x u_y + c_1)(2\sigma_{xy} + c_2)}{(u_x^2 + u_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.3}$$

where, signal x and y are the original and the distorted frame information, respectively. u_x, u_y refer to the mean of the x and y, respectively. σ_x, σ_y and σ_{xy} denote the variance of x, the variance of y and the covariance between x and y. c_1, c_2 are constants (c) to avoid values of $\mu_x^2 + \mu_x^2$ and $\sigma_x^2 + \sigma_x^2$ very close to zero, respectively. SSIM has a significant advantage in the motionless frames or changed slightly frames since it considers not only the physical difference of the pixels but also the structure distortions. However in high motion sequences it is not very accurate.

The NR VQA for both 2D and 3D video in real time applications requires objective metrics. The objective video quality metrics are mainly validated by the Video Quality Experts Group (VQEG) which results in International Telecommunications Union (ITU) recommendations and standards for objective quality models. In accordance with ITU standards objective VQA metrics can be classified according to the type of input parameters as packet-layer models, bitstream-level model, media layer model, and hybrid model [20], as depicted in Figure 2.1.

As input information in packet layer models, several different types of parameters can be used, such as, packet headers (e.g., RTP header, TS header), network parameters (e.g., packet loss rate (PLR) , delay, jitter), and also codec configuration information. On the one hand, the video signal is not available at this level, and this type of model cannot precisely locate the impaired parts. On the other hand, this type of model is suitable for real-time applications considering the low computation complexity required to extract high level information from the video stream. As depicted in Figure 2.1,

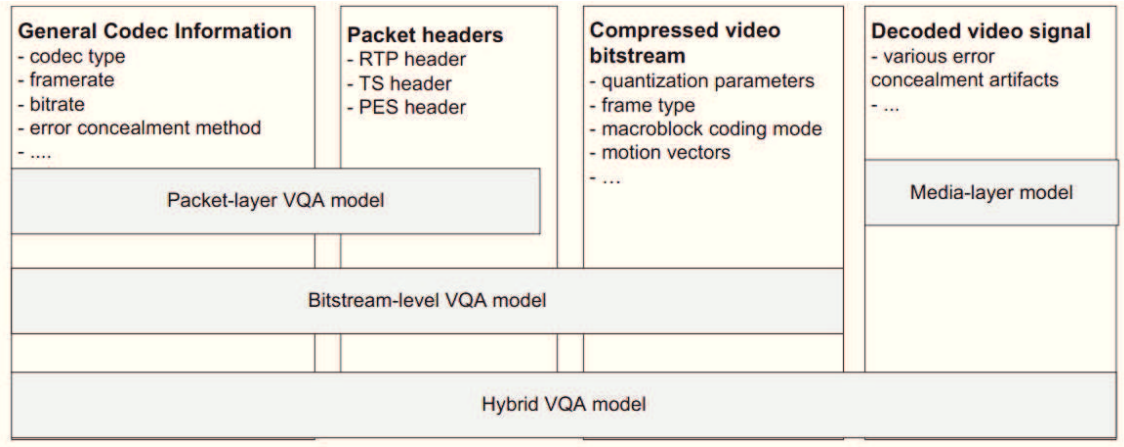


Figure 2.1: Scope of the four types of VQA models [21].

the bitstream-level model also uses information from the compressed video bitstream in addition to that of the packet headers. Hence, this type of model has access to the encoding parameters, from which it is easier to determine the location of the impaired parts of the video signal. Bitstream-level models have higher computational complexity than the packet-layer model, but it is also more accurate. However, when the video bitstream is encrypted in general it is not possible to use these type of models, because the stream cannot be decoded in the middle of the network. In the case of media-layer models, they operate at pixel level and so, content-dependent features that influence video quality can be detected and evaluated (e.g., texture masking effects and motion masking effects). However, if packet loss information is not used, that it is more difficult to detect the distorted parts in the distorted video. As the name suggests an hybrid model combines several features of the previous type of models. It employs the decoded signal itself in addition to the bitstream and packet-header information to improve video quality prediction. Considering the amount of information used in this model it can provide the most accurate quality prediction performance. However, as expected, it is also much more complex in terms of computational requirements.

Since the goal of this work is to monitoring the 3D video quality at network nodes of the transmission networks, the NR packet-layer model is the VQA model that better fulfills all the needs in the scenario under study. To understand how this type of models work, a literature review was made in order to support the research. Although several quality models were developed for two-dimensional (2D) video streams [21, 22], the case of 3D video streams is not yet fully investigated.

This chapter contains a state-of-the-art concepts review related to Video Quality Assessment (VQA) methods for both 2D and 3D video content. These methods are analysed

to find out which one better suits this work's goals and specifications. Several quality models are presented for 2D and 3D video content in Sections 2.1 and 2.2. To understand how they work and which contribution they might provide to this study is the main goal of the chapter.

2.1 2D video quality models

Regarding the impact of packet-loss and transmission errors in the video content, the QoE reduction in function of the impaired section of the frame, and how it propagates due to the temporal prediction was under study in [23]. RTP (X) packet-header specification was used, since it allows the usage of a packet-header extension. Thus, an additional structure and corresponding elements was defined by the authors, as depicted in Figure 2.2. The first 32-bit word contains a profile-specific identifier (16 bits) and a length specifier (16 bits) that indicates the length of the extension in 32-bit units, excluding the 32 bits of the extension header.

The method proposed in [23] consists in measuring the impaired section resulting the packet-loss events. Therefore, a frame with an impaired level of 50%, means that half of the frame was damaged by the packet-loss. In the first place it is determined the level of the frame degradation and then the error propagation is measure (e.g., impairment propagation to the upcoming frames).

To determine the degradation level the H.264/AVC slice header information was taken into account, such as slice type and reference frames buffer indexes. The degradation level is measured in a scale from 0 to 100, where 0 means that the frame was successfully delivered, and 100 when it is completely loss. This model determines the impaired frame level as follows:

$$E_0 = 100\frac{1}{N}\sum_{S=0}^{N-1} 1 - f\left(\frac{L(S)}{L_{avg}}\right), \tag{2.4}$$

where:

S : identifies each slice in one frame.

N : number of slices in one frame.

$L(S)$: is the non-lost slice size, in bytes.

L_{avg}**:** estimated slice size if there were no losses.

Although the decoding process of H.264/AVC includes both temporal and spatial prediction, in this case only temporal prediction is taken in to account since this the one responsible for error propagation between adjacent frames. So, when the objective

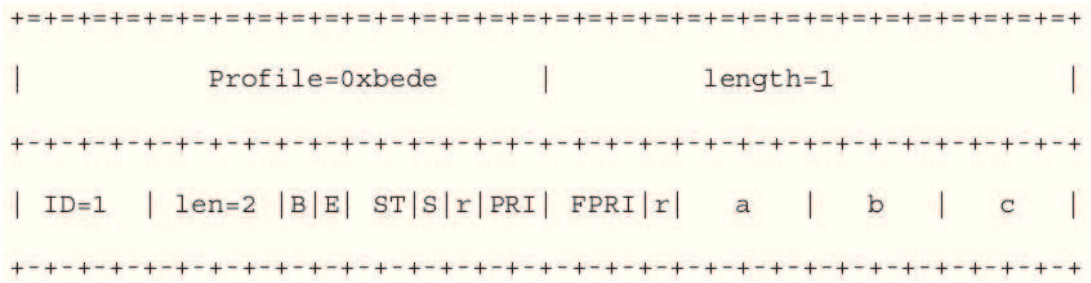


Figure 2.2: RTP packet-header extension [23].

is to determine the frame degradation, several issues mus be considered. For instance, not only the percentage of the impaired frame itself (E_0), but also the propagation of that impairment to the adjacent frames which rely upon this one (E_p). Given a frame x, depending on a set of references yk, propagated error will be:

$$E_p = \gamma \sum_k \omega_k E(y_k), \tag{2.5}$$

where:

$E(y_k)$**:** is the error level in the frame y_k.

$\gamma < 1$**:** constant that is dependent with the error attenuation along the reference frames.

ω_k**:** is a set of weight values associated with each of the reference images. It is used a model where higher-level errors have a higher weight, as they propagate in a more perceptible way. The model is:

$$w_k = \frac{E(y_k)}{\sum_k E(y_k)}, \tag{2.6}$$

In the best case scenario, i.e., when there is no propagation of the E_0 error, E_p and E_0 will overlap and the final error will be maximum of them. In the worst case scenario, when there is propagation of the E_0 error, the E_p and E_0 are independent and final error will be sum of them.

This model presents high correlation with the actual distortion when the error occurs in the frame with the impaired area. If the error event occurs in the beginning of the coded frame, then the impaired area in that frame will be bigger when intra prediction is used. This model can be important for this work, since it allows to determine the impact of packet-loss events in the perceived video quality. And as it can be seen in the model equations, almost all parameters employed in this model can be adapted and used in 3D video streaming scenarios.

Video system planning and in-service video quality monitoring were the scenarios set side by side in [21], where it is concluded that the Packet Loss Rate (PLR), Burst

Loss Frequency (BLF) and Invalid Frame Rate (IFR) used in video system planning are very effective, while for video monitoring systems they are not so effective. A 2D video quality monitoring is also proposed in [21], which uses the spatio-temporal complexity to extrapolate the video quality. This model connects the video content features with the effectiveness of Error Concealment (EC) techniques, and also with the propagation error effects. The visible artifacts duration and annoyance level caused in the costumer are highly correlated with the perceived video quality. It can be seen high dependency between the annoyance level and the error concealment, since error concealment reduces the impact of lost information in the impaired video stream with the goal of minimising the perceived video quality degradation. The inter-frame prediction used in video coding brings out several temporal dependencies levels, between frames. During packet-loss events, this dependency is reproduced in a propagation error which is responsible for visible artifacts during loss periods. In general, a packet-loss event affecting an I-frame leads to an artifact with a life-time of a Group of Pictures (GOP). If an open GOP structure is used, then the length distortion period can be higher. Typically, a P-frame artifact life-time is smaller than a I-frame. However, when there is a lot of motion in the adjacent packets of the P-frame ones, then losing one of those can have significant impact. For B-frames, error propagation does not happen. Both error concealment and error propagation are directly related with the spatio-temporal complexity, of the adjacent lost packet and this is the point exploited by this model.

The RTP packet-header sequence number field is used to detect packet-loss events in a video sequence. To identify the different frame types, the timestamp field is used. Historic frame size statistical evaluation is used to estimate the lost frame size. Experimental results show that video monitoring method with input parameters packet-header information and coding configurations, is able to determine video quality with the accuracy expected in real scenario. The work described in [21] contains useful information for 3D video quality monitor deployment, since error concealment techniques and spatio-temporal dependencies are also used in the 3D streaming. Besides the spatio-temporal complexity, 3D video coding also takes advantage of the redundancy between views (i.e., inter-view prediction in H.264/AVC). Despite the unclear model implementation and results not very well detailed, the framework and approach used are interesting in the scope of this dissertation.

One of the methods described in [22] is called No Parse and only uses network level information, and thus without access to the specific information of the video sequence, such as spatio-temporal activities and video signal error location. This method estimates Mean Square Error (MSE) using PLR information. There are several issues associated with this method, which is a very simple model when compared with the Quick Parse and Full Parse methods, also presented by the same author. Firstly, No Parse only uses

linear approximation in multiple sequences, which is not so much accurate even when the sequences are coded with the same configuration parameters, thus the video signal has an high and unpredictable complexity. Secondly, linear approximation is not accurate when there is multiple adjacente frame loss. Thirdly, even in the single-loss scenario with low PLR, the same video sequence can have high MSE variance. Considering these topics, measuring video quality using only packet-loss rate information cannot represent the perceived sequence video quality with reliability. However, there are scenarios where the bitrate and PLR are the only available information (e.g., encrypted bitstreams). In this case, methods such as No Parse are very useful for video quality monitoring. This could be the starting point for a 3D video quality monitor when no access to the compressed bitstream is provided, and it could be optimized according to the amount of available information.

A packet based parametric model which allows monitoring the video quality of IPTV is presented in [24] and [25]. The packet header information is the input parameters, as depicted in Figure 2.3. One cannot find the relationship between the packet content and video quality, because decoded video stream is not available at network level. The conducted experimental performance measures concludes that subjective video quality degradation increases with the amount of packet-loss. Thus, the video quality (V_q) exponentially decreases with the increasing of Packet Loss Frequency (PLF) , i.e., number of lost packets within 10 seconds interval. The experimental tests led to a mathematical model that determines the video quality as described by the following expression:

$$V_q = 1 + I_c exp\left(-\frac{PLF}{v_4}\right), \tag{2.7}$$

where $1 + I_c$ is the video quality when the PLF is zero and v_4 stands for video degradation level due to packet-loss. Degradation level due to packet-loss is constant for a given bitrate, thus v_4 represents degradation level due to packet-loss for the highest experimentally coded bitrate. This model was validated with correlation results above 0.9, between the subjective tests and the estimated video quality. This model can be useful for this work, since it allows to measure 2D video quality in a packet-loss scenario. Furthermore, it will be explained how 3D video quality monitoring is related with 2D video quality.

Several video artifacts like blocking or pixelization, ghosting and frame freeze are linked to video sequence QoE, during packet-loss events, in [26]. Frame size differs with the amount of motion and frame content complexity, and thus it is concluded that the motion complexity variation influence the number of impaired frames. The lesser the motion information, the bigger the amount of impaired frames, hence the visible artifact will be higher. Impairment duration is also affected by the GOP size, i.e., the bigger the GOP size, the bigger the error propagation. Given the following GOP structure $x : y$, at z

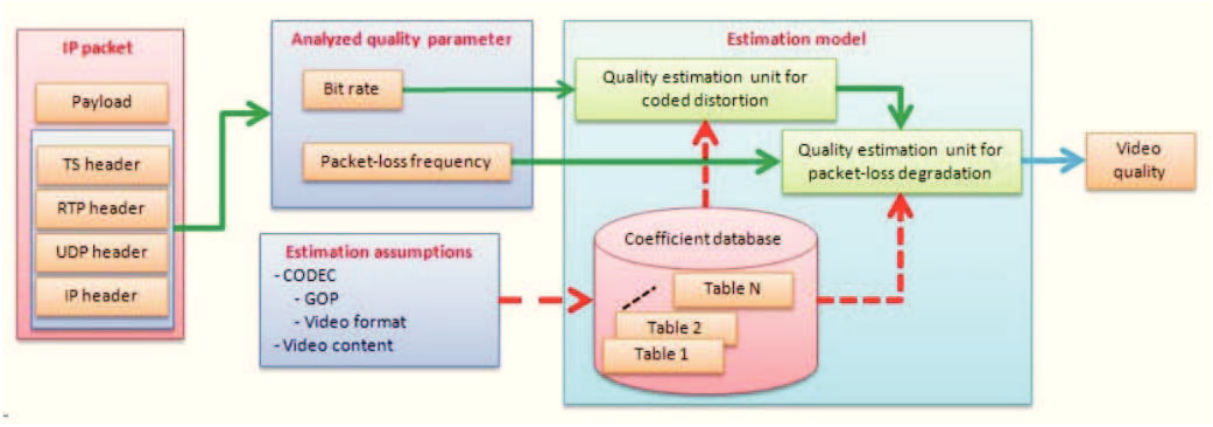


Figure 2.3: IPTV video quality monitoring model [25] and [24].

frames per second (fps) , in the worst scenario $(x+y)\ \frac{(x+y)}{z}$ seconds will be the maximum error length. Even with previous errors, the video quality is restored when an I-frame is properly received. Since this work links the frame size with the amount of motion, and hence with video quality, the inherent conclusions are relevant for the development of a NR video quality monitor. Also important is the amount of motion estimation using frame size fields, which is available at packet headers level.

The characterization of the distortion introduced by the transmission channel is another way to assess video quality. There are several models in the literature that characterize the distortion introduced by the channel, for 2D video. For example, considering low complexity models, the distortion inserted in the transmission process can be analyzed as intra coding type and spatial-filtering, as described in [27]. A GOP based distortion model, that analyzes the frame error propagation behaviour is proposed in [28]. The proposed model in [29] uses intra and inter-frame prediction and deblocking-filtering to determine MSE channel distortion information.

2.2 3D video quality models

Despite the amount of literature addressing transmission channel distortion models for 2D video, there is still a lot of open issues and work to be done for 3D video .

Error concealment techniques and spatio-temporal complexity (e.g., intra-frame and inter-frame prediction) present in 2D video coding algorithms are also occurring in 3D scenario. Since these are key topics in video quality evaluation, one can conclude that 3D video quality models can be seen as an extension of existing models for 2D video. This will be the line of thought of this work, where the proposed 3D video quality monitor will rely upon the models referred in the last section.

However, in 3D world it is necessary take into account other aspects besides the ones

present in 2D video, such as inter-view prediction and 3D binocular artifacts (e.g., occlusions, perspective distortions, depth distortions and the detection of binocular rivalry). Figure 2.4 shows the block diagram of a typical transmission chain. Several blocks are common to a 2D and 3D transmission chain but, in most cases, additional processing steps are required for 3D, and the delivery of 3D signals causes new types of artifacts. Thus, when compared with 2D video, 3D video quality assessment becomes much more complex as depicted in Figure 2.4 and described next [30]:

- there are additional steps in the transmission chain that need to be addressed;
- the observer's opinion may be considered as multidimensional, including factors like visual fatigue and depth perception;
- more aspects of the Human Visual System (HVS) need to be addressed, e.g. binocular rivalry, binocular suppression.

These features need to be taken into account when monitoring objective 3D video quality. For this purpose, a rate distortion optimization process for error resilience of video coded with inter-view prediction was developed in [31]. A model that estimates the distortion caused by H.264/Multiview Video Coding (MVC) video packet loss distortion is presented in [32]. In 2D video the error propagation happens only in one dimension (temporal intra-view), however for multiview video the error propagates through the base-view reference frames but also through the auxiliar view frames, and thus it is a two-dimension propagation (i.e., temporal and inter-view prediction).

The possibility of predicting 3D video quality using 2D quality metrics in packet loss conditions for both left-and-right and colour-and-depth stereoscopic videos, is addressed in [33]. The relationship between subjective quality measures and several objective quality measures like PSNR, SSIM, and VQM for 3D video content are also investigated. Research is being carried out covering the whole chain of 3D video from 3D capture to 3D display technologies, as depicted in Figure 2.5. However, the effect of these technologies on the perceptual aspects of 3D viewing has not been thoroughly investigated to date. Even

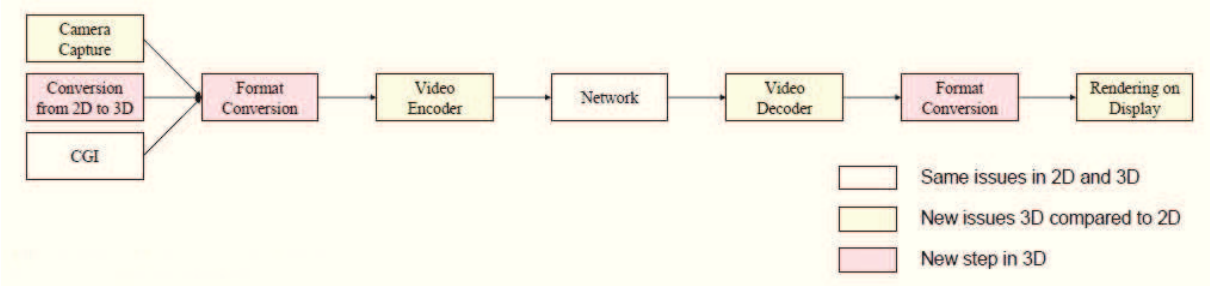


Figure 2.4: 3D video transmission chain [30].

though there are 2D objective quality models which are highly correlated with Human Visual System (HVS), very little work has been carried out to find objective measurement techniques for 3D video quality. Results show that the average VQM score of the left and right video demonstrate similar trends compared to perceptual 3D quality scores. Furthermore, it has been noted that the raw MOS scores from VQM are consistently worse than viewer ratings. This suggests that the addition of the depth perception increases the viewer satisfaction significantly, masking the effects of compression and transmission impairments in the 3D video.

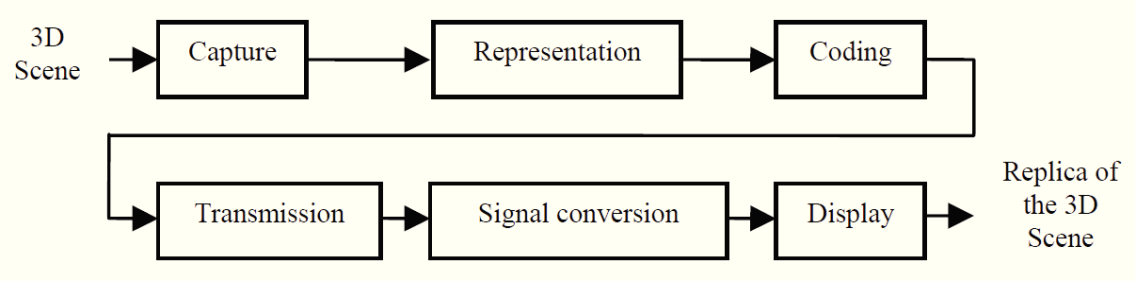


Figure 2.5: 3D video transmission chain [33].

The model proposed in [32] has the ability to deal with the two dimensions of the propagation errors. This is a recursive model, which iteratively determines the current frame distortion, using both views and previous frames. The theoretical model results were compared with the experimental ones, in both subjective and objective evaluation. The results show that the model determines the transmission channel distortions with high accuracy, for both frame and Group-of-Pictures (GGoP) level. Video quality assessment is directly connected to channel transmission distortion, thus to evaluate the objective quality of a video stream it is necessary to characterize the channel regarding the distortion inserted in the stream. Therefore, the model presented in [32] is important to systems that need to evaluate 3D video quality, since it characterizes the distortion in a H.264/MVC video transmission channel.

A NR quality assessment algorithm for 3D video and images is described in [34]. For images, the algorithm extracts statistical disparity features, as well as spatial activity indicators. In video scenario, the algorithm determines the QoE using the image spatial activity indicators together with the set of motion compensation disparity differences. Using a set of QoE 3D data available in [35] and [36], firstly it was shown that the proposed algorithm correlates well with the user's perception of quality. Secondly it was observed that the comfort related to the stereoscopic view, decreases with the distance to the reference point. And thirdly it is observed that this phenomenon pronounces more high disparity indoor scenes.

Stereo video quality evaluation using 2D objective quality models is the goal of the work presented in [37]. A good correlation between colour plus depth subjective and objective video quality correlation was found by the authors. The subjective tests were made to determine the image quality and the depth perception for a set of asymmetric coded video sequences. The results show that the video quality determined objective metrics such as Video Quality Metric (VQM) is highly correlated with the users perceived depth and image quality.

In multi-view coding, the bit rate for the right eye view is lower than that for the left eye view because the right view is encoded using inter-view technology. In addition, the right view can be encoded at a much lower bit rate on the basis of binocular suppression. Thus, the video quality for the left view can be different from that for the right view. The work in [38] explores how such a video quality difference between the left and right views affects the overall 3D video quality. To derive perceived quality characteristics, subjective quality tests were conducted. It was concluded that the difference in quality between the left and right views has little influence on the overall 3D video quality and that the overall 3D video quality can be modeled using the 2D video quality for left and right views. They also show that the subjective video quality can be modeled by a multiple regression function, where the independent values are the image quality for the left and right views.

A Reduced-Reference quality metric for 3D depth maps transmission based on edge information is presented in [39]. The quality metric process using edge information as side-information is shown in Figure 2.6. Initially, side-information (i.e., edge information) is generated from the original depth map and then transmitted over the Reduced-Reference channel to the receiver. Ideally, this RR channel should be lossless. In the case of in-band transmission of side information, a high protection through unequal error protection can be provided. At the receiver- side the edge information is also obtained from the processed/received depth map. Then these two binary edge masks are compared to obtain an index for the structural degradation of the depth map. In addition to this structural comparison, contrast and luminance comparisons are performed based on the statistics received from the sender-side. The final quality rating will reflect luminance, contrast and structural degradations of the corrupted depth map with respect to its original depth map sequence. An Edge-based SSIM metric is proposed in [40]. This metric presented in eq. 2.8 is used to integrate luminance, contrast and structural information into the metric.

$$EB - SSIM = [l(x,y)]^{\alpha}.[c(x,y)]^{\beta}.[s_e(x,y)]^{\sigma} \tag{2.8}$$

where $S_e(x,y)$ stands for structural comparison performed using the edge information

generated from the original and processed depth maps. Results of the proposed method show a good approximation of the Full-Reference quality metric for all the considered sequence types, PLRs and compression levels. This suggests that due to the practical problems associated with Full-Reference methods (i.e., need for bandwidth), a Reduced-Reference quality metric as described in this paper is an acceptable compromise for the 3D video research and development community. However, RR metrics need a side-channel information which in real-time applications might be impossible, and thus the only way to assess the stream video quality is to use NR quality metrics.

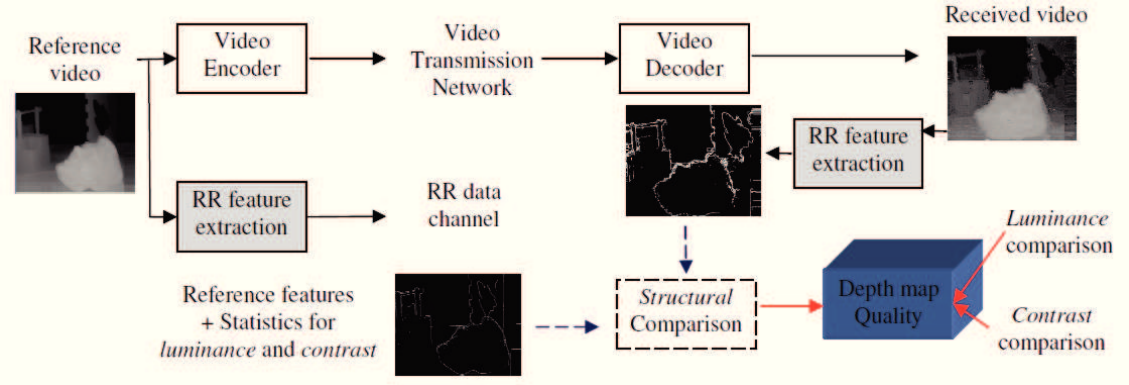


Figure 2.6: Block diagram of the proposed Reduced-Reference quality metric [39].

Factors that affect human perception of depth and visual comfort from stereoscopic video were studied in [41]. Subjective quality assessment tests were conducted to extract four factors: temporal variance, disparity variation in intra-frames, disparity variations in inter-frames and disparity distribution of frame boundary areas. A no-reference stereoscopic video quality perception model (SV-QPM) was designed based on these four factors. The SV-QPM does not require the depth map, but utilize the disparity information by simple estimation, and the model parameters are estimated based on linear regression. The experimental results show that model proposed in [41] exhibits high consistency with subjective quality assessment results in terms of a Pearson correlation coefficient value of 0.808, and the prediction performance exhibits good consistency with zero outlier ratio value.

A no-reference objective quality measure for stereoscopic 3D videos generated by depth image based rendering (DIBR) was presented in [42]. As a first step, an ideal depth estimate was derived for each pixel value, which was used to calculate three distortion measures: temporal outliers (TO), temporal inconsistencies (TI), and spatial outliers (SO). The proposed no-reference measure is composed by the combination of the three measures. DIBR has many advantages over two-views approach including high bandwidth efficiency, user interactivity, computational and cost efficiency, and 2D to 3D selectivity

[43]. DIBR also eliminates photometric asymmetries in between the two views, hence both of them are generated from the same original image. Figure 2.7 depicts the factors that affect the the perceived quality of DIBR-based stereoscopic 3D videos:

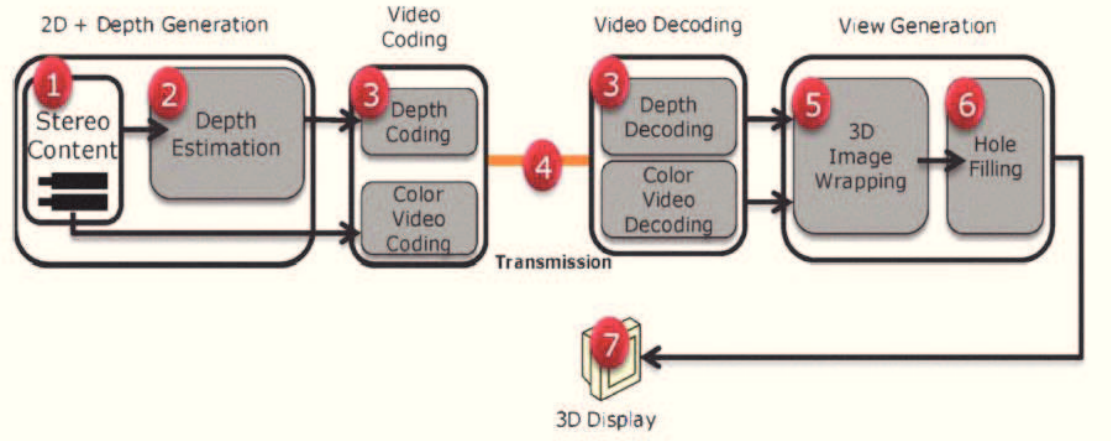


Figure 2.7: The perceived quality of the DIBR video is affected by every block in the processing chain: (1) Video Capture, (2) Depth Estimation, (3) Coding, (4) Transmission, (5) 3D Wrapping, (6) Hole Filling, and (7) Display scaling and formatting [42].

- accuracy of the estimated depth maps;
- quality of the 3D wrapping process in DIBR;
- quality of the hole-filling algorithm applied to cover the disoccluded areas in the generated frames;
- compression artifacts for the 2D video and depth map;
- transmission errors and streaming losses;
- scaling and formatting algorithms in the 3D displays;

While 2D video quality is solely based on monocular color cues in one view, 3D video quality on the other hand is a combination of binocular and monocular cues. In DIBR, virtual views are generated by first projecting the pixels in the reference image to the world coordinates using depth map and camera information. The resulting pixels in the world coordinates are then sampled in the 2D plane from a different view-point to obtain a DIBR estimated image. In Figure 2.8 one can see the reference camera C_r and virtual camera C_v. Where F_r and F_v are the focal lengths of both cameras.Z_c is the convergence distance of two cameras. In [42] three measures were combined (e.g., temporal outliers

(TO), temporal inconsistencies (TI), and spatial outliers (SO)) into one no-reference 3D vision-based quality measure for stereoscopic DIBR-based videos as follows:

$$NR-3VQM = K(1-SO(SO \cap TO))^a(1-TI)^b(1-TO)^c \tag{2.9}$$

where SO, TO, and TI are normalized to the range 0 to 1 and a, b, and c are constants which were determined by running a few training sequences. $(SO \cap TO)$ is the logical intersection of SO and TO included in the equation to avoid accounting the outlier distortion more than once. K is a constant for scaling where NR-3VQM ranges from, 0 for lowest quality, to K, for highest quality. The overall quality measure is calculated as the mean of the values in the matrix NR-3VQM.

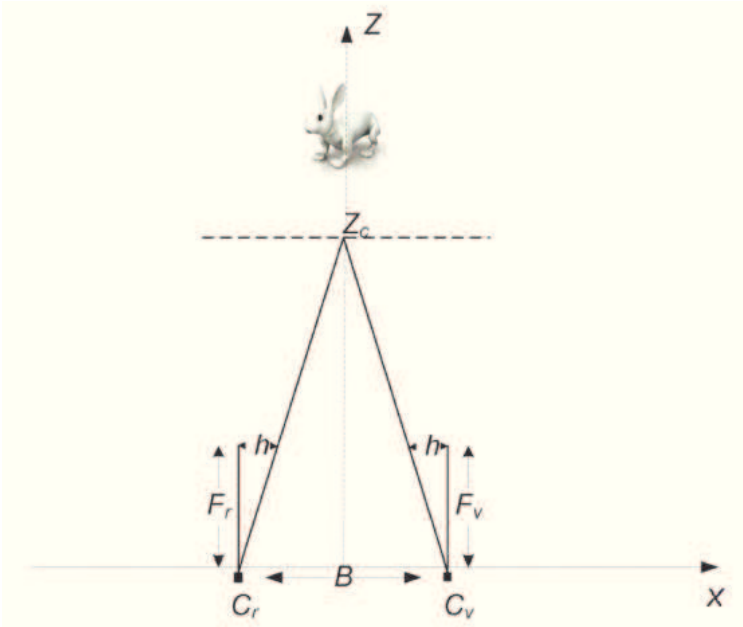


Figure 2.8: Depth image-based rendering (DIBR) [42].

The performance of the NR measure proposed in [42] was verified using subjective DMOS scores and compared to the full reference version of the proposed algorithm. The results have shown that the predictions of the no-reference measure highly correlates with subjective scores and is fairly close in performance to the full reference 3VQM.

A no-reference objective color video quality assessment metric was presented in [44]. It was defined a flow tensor between successive frames that is weighted by the perceptual mask and used to define a NR color video quality metric. The inter-frame coherence and the sharpness of edges in the successive frames was studied. Experiments performed on video sequences indicate that the objective scores obtained by the proposed metric agree

well with the subjective assessment scores.

The relationship between the perceptual quality of stereoscopic images and visual information was explored in [45], where the authors also proposed a no-reference quality metric for stereoscopic images based on a model for binocular perception. This metric is a top-down method, modeling the binocular quality perception of the human visual system in the context of blurriness and blockiness. Perceptual blurriness and blockiness scores of left and right images were computed using local blurriness, blockiness, and visual saliency information and then combined into an overall quality index using the binocular quality perception model. Experiments for image and video databases show that the proposed metric provides consistent correlations with subjective quality scores. The results also show that the proposed metric provides higher performance than existing FR methods even though the proposed method is based on an NR approach.

Objective no-reference metrics for automatic color- and sharpness-mismatch detection in video captured using stereo cameras were introduced in [46]. View matching and reconstruction are the features in the algorithms base. It was proposed a fast block-based color-independent algorithm for stereo matching. The presented quality-assessment procedure reveals scenes distorted during film production or postproduction and enables film comparison in terms of stereoscopic quality. For NR video quality assessment applications, the PSNR values can precisely be predicted only if the parameters of distortion models are estimated. The architecture of no-reference PSNR estimation is shown in Figure 2.9. The PSNR for n-th frame f_n can be predicted by the model without any references in the decoder side. In order to predict the model parameters of the Laplacian mixture distribution for all zero quantized coefficients case, an exponential regression scheme was employed over quadtree depth levels of CUs. The proposed no-reference PSNR estimation method yields fairly accurate results from 0.970 to 0.983 in correlation and from 0.530 to 0.890 in RMSE between the actual and the estimated PSNR values for HEVC test model

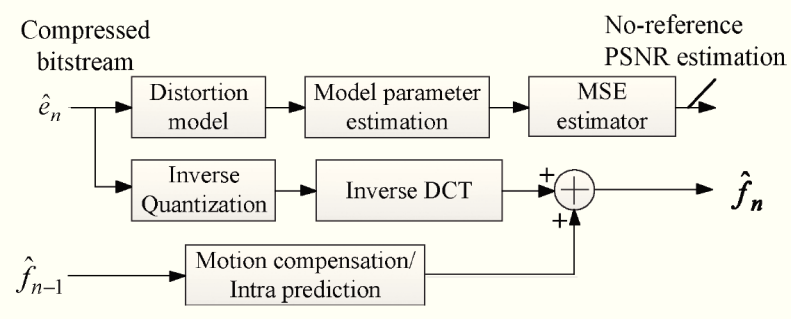


Figure 2.9: Architecture of no-reference PSNR estimation in decoder side, proposed in [46].

(HM) encoded bitstreams, outperforming single probability density function (PDF) based models.

A no-reference PSNR estimation method was presented in [47]. The presented no-reference PSNR estimation method is based on a Laplacian mixture distribution, which takes into account the distribution characteristics of residual transform coefficients in different quadtree depths and coding types of coding units (CUs). The authors in [48] present an 3D NR algorithm that predicts the quality of stereo images whether the distortion is symmetric or not (rivalrous). The algorithm extracts both 2D and 3D natural statistical features from a stereopair to evaluate its quality. A distorted stereopair is first classified into symmetrically os asymmetrically distorted using these features. In Figure 2.10 is shown a flowchart of the proposed model in [48]. Given a stereo image pair, an estimated disparity map is generated by a Structural similarity (SSIM) based stereo algorithm, while a set of multi-scale Gabor filter responses are generated on the stereo images using a filter bank. A Cyclopean Image is then synthesized from the stereo image pair, the estimated disparity map, and the Gabor filter responses. A cyclopean image is defined as

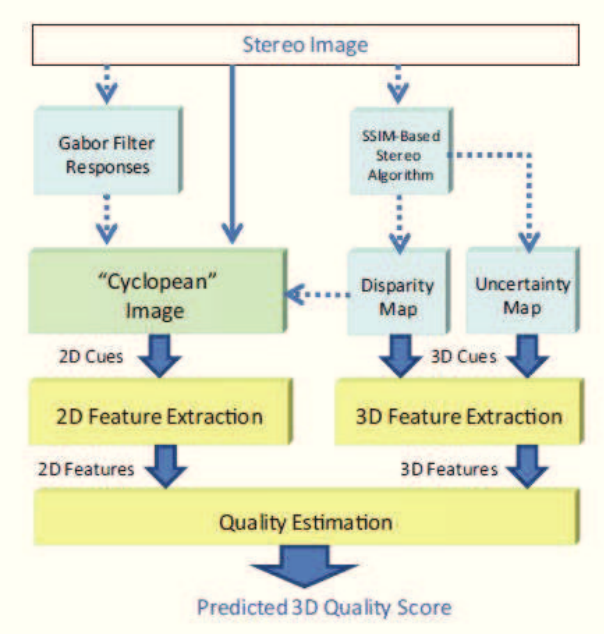


Figure 2.10: 3D NR quality model flowchart proposed in [48].

the average of the left image and the disparity-compensated right image [49]. Of course, the Cyclopean image is not an average of the left and right views but instead, is a 3D percept consisting of luminance and color patterns superimposed on a 3D images [50], [51]. Synthesized Cyclopean Image was used to extract 2D features, while 3D features are independently extracted from the estimated disparity map and an uncertainty map that is also produced by the stereo matching algorithm. Finally, the perceived 3D quality of each tested stereo imagepair is predicted once the quality estimation module is fed with the

2D and 3D features. This is a quality model for 2D and 3D images, however video quality features can be derived from the same algorithm, since the 3D video has also 2D video characteristics. This is an interesting model since it is a SSIM-based algorithm which is an accurate metric to determine the objective quality.

2.3 Summary

This chapter presented a review of the different types of VQA models considering the type of input information. Video quality models and how they are relevant for 3D video monitoring were discussed in section 2.1. In section 2.2 several 3D VQA models presented in the literature were reviewed. One can conclude that spatio-temporal complexity presented in 2D video coding algorithm also have significant impact on the case of 3D video. Based on the models discussed in this chapter a 3D video quality model is proposed in chapter 3 where an overview of the model is presented, followed by the implementation and discussion of results.

Chapter 3

Proposed NR packet-layer model

This Chapter presents a 3D video quality model that quantifies the objective video quality in the presence of frame losses without decoding the compressed stream. This video quality monitor only requires information obtained from the transmission network such as the packet losses and the raw packetised stream itself. Such system is able to operate at any point of the communication path. In the next section a possible networking scenario is described, followed by the details of the proposed model. Section 3.3 presents the model implementation and the simulation results are presented and discussed in Section 3.4. A summary of this Chapter is presented in Section 4.5.

3.1 Network scenario

Nowadays there are different options to deliver 3D video over networks. In regard to this work, it is of particular interest to distinguish between broadcasting systems, which always use MPEG-2 Transport Streams (TS), and transmission over the Internet and IP based networks, which might not use TS [52].

This specification is constrained to the case of IP networks using cable/fiber which do not necessarily need TS encapsulation. Therefore, since the use of TS over RTP introduces overhead redundancy, the option considered was the broadcast systems. Taking into account the video communication layered stack, the Video Coding Layer (VCL) is followed by the Network Adaptation Layer (NAL) and then by the network layers using the Packetised Elementary Stream (PES), MPEG2-Transport Stream (TS) and Internet Protocol (IP). Thus, Figure 3.1 depicts the specification that conforms with the following protocol stack: VCL/NAL/PES/TS/IP.

The network scenario is set, and thus the next step is to understand which network parameters can help to determine the video quality based upon the information collected

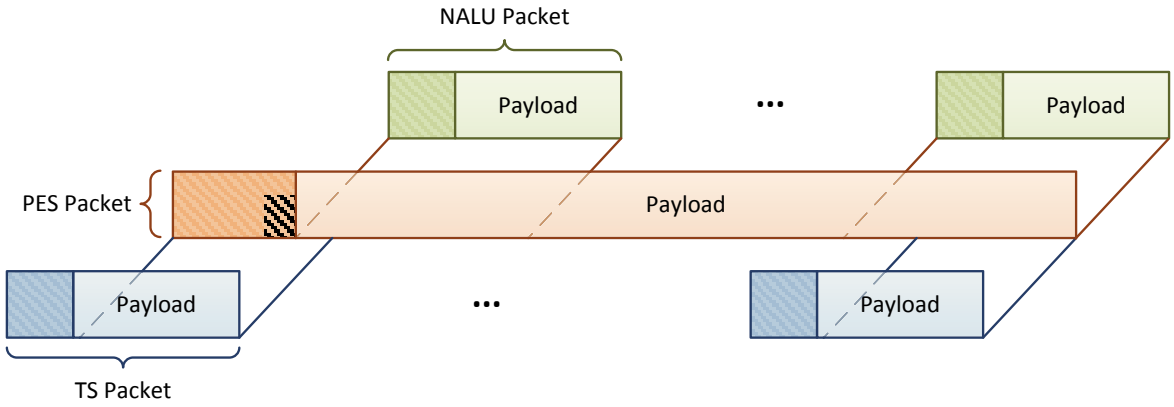


Figure 3.1: Network stack scenario.

from packet-loss events. A brief description of the protocol stack packet's header information is presented next.

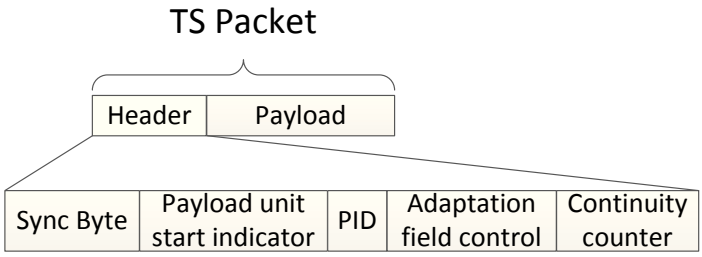


Figure 3.2: MPEG2-Transport Stream packet header parameters.

- **Synchronisation Byte:** used to determine the begin of a TS packet;
- **Payload Unit Start Indicator:** indicates if the TS packet payload transports the begin of PES data;
- **PID:** packet ID, identify the TS packet;
- **Adaptation field control:** indicates if the packet header contains the Adaptation Field (AF) ;
- **Continuity counter:** TS packet number counter, incremented only when a payload is present;
- **Packet start code prefix:** used to determine the begin of a PES packet;
- **PES packet length:** indicates the size of the data in PES packet payload;

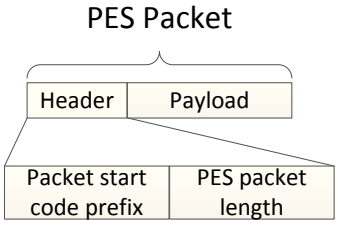


Figure 3.3: Packetised Elementary Stream packet header parameters.

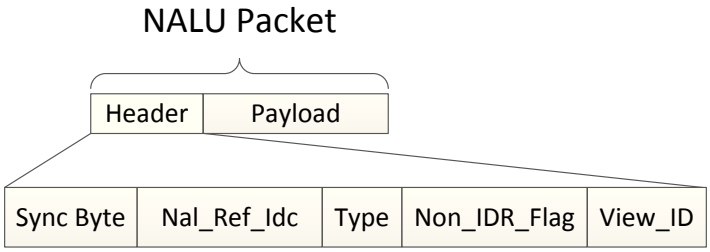


Figure 3.4: Network Abstraction Layer packet header parameters.

- **Synchronisation Byte:** used to determine the begin of a NAL packet;
- **NAL_Ref_Idc:** indicates that the content of the NAL unit is used to reconstruct reference pictures for inter picture prediction.;
- **NAL_Unit_Type:** this component specifies the NAL unit payload type;
- **Non_IDR_Flag:** indicates if the NAL data payload transports IDR-frame content;
- **View_ID:** specifies which view does the NAL packet belong, only in a multi-view scenario;

3.2 Quality model overview

As introduced in Section 3.1, a broadcast network is assumed, which corresponds to a protocol stack with TS/PES/NAL. These are the packet layers used in the proposed model as depicted in Figure 3.5, where the data units from the Video Coding Layer (VCL) are wrapped into NAL packets which in turn make up the PES payload, and finally the whole PES packet is split into TS packets.

The proposed model aims to estimate the quality degradation in isolated stereo frames due to errors in TS packets leading to frame loss. The degradation is measured by the difference between the SSIM of the error-free stereo frame (i.e., SSIM=1) and that of the displayed frame, assuming that frame-copy is used as the concealment method. Note that coding distortion is not included in this model, which means that SSIM=1 for any

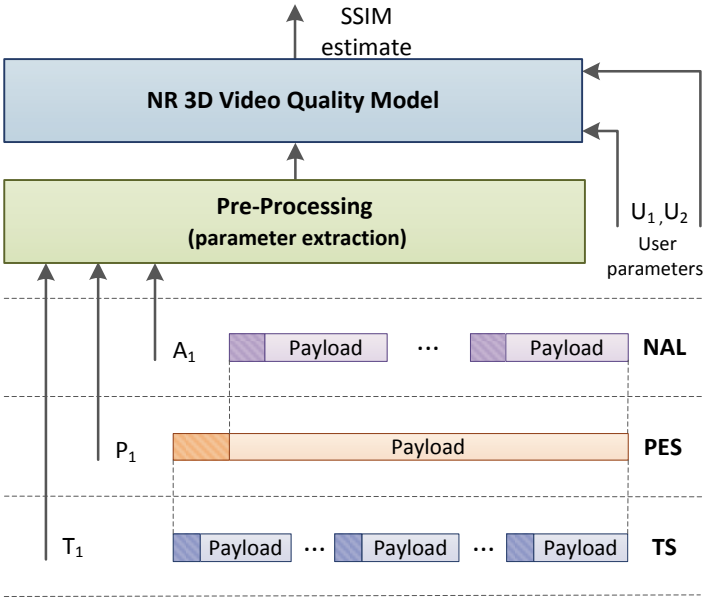


Figure 3.5: Packet layer model structure

frame correctly received, thus decoupling the errors due to packet losses from the loss of fidelity due to lossy encoding. The proposed NR model uses an estimate of each lost frame size and different parameters for different frame types. The lost frame size is estimated from the average of the last frame sizes with the same type and view. The GOP structure is provided as input parameter, since this type of information is in general static. Nevertheless, estimation of different frame types can be done with reasonable accuracy even without knowing the GOP structure. The detection of lost frames is based on the following parameters extracted from the packet headers:

- **T1: Continuity Counter (TS)**: enables detection of TS packet loss events.
- **P1: PES packet length**: provides the frame size.
- **A1: NAL_unit_type**: identifies the frame type.
- **U1: GOP size**: GOP size information.
- **U2: GOP structure**: GOP structure information.

3.3 Proposed NR model

Several experiments were performed to collect relevant statistical data on the impact of frame loss in 3D video streams quality. The experimental procedure have five stages (e.g., 1-H.264/AVC coding sequence, 2-MPEG2 – TS encapsulation, 3-Packet loss events, 4-H.264/AVC decoding process and 5-SSIM computation.) as illustrated in Figure 3.6. To

conduct these experiments a 3D video sequence obtained by concatenating 5 individual sequences (Ballons 3.7(a), Champagne Tower 3.7(b), Kendo 3.7(c), Pantonime 3.7(d) and Dog 3.7(e)) was encoded with H.264/AVC Stereo High Profile using the reference software JM 18.2 (Figure 3.6 stage 1). This sequence has spatial resolution of 1024x768 pixels, 30 fps frame rate, and was encoded using GOP size equal to 21 frames and IBPBP GOP structure. Three different datasets were created by encoding this sequence with QP ranging from 26 to 32, achieving different PSNR and bitrates. These datasets are described in Table 3.1. The resulting video stream was encapsulated into a TS stream using the reference software FFMPEG (Figure 3.6 stage 2).

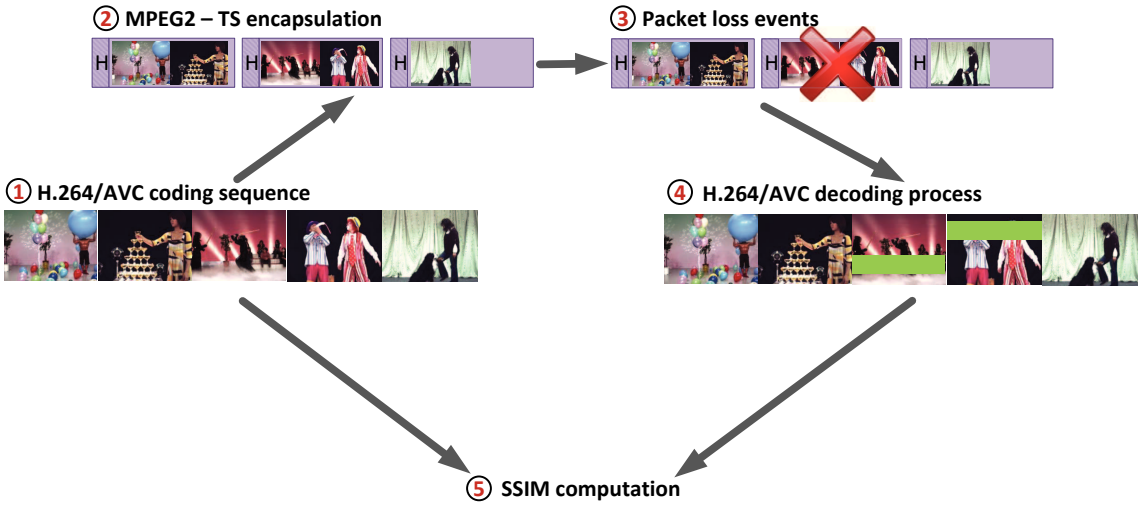


Figure 3.6: Experimental procedure.

To simulate network loss conditions, packetized streams were then subject to packet loss events (Figure 3.6 stage 3). These have to be created according to a error pattern, produced in a controlled manner, to better modeling individual types of impairments. Aiming this, at present a single frame is lost in each loss event, but other impairment models can be used [53].

Table 3.1: QP's, PSNR and bitrate.

	I-QP	P-QP	B-QP	PSNR (dB)	Bitrate (Kb/s)
Dataset-1	26	28	28	42	2847
Dataset-2	28	30	30	41	2225
Dataset-3	30	32	32	40	1727

Then, the corrupted TS stream was decoded using frame-copy for concealment of lost frames, as depicted in stage 4 of Figure 3.6. The decoded frames were used to

(a) Ballons.

(b) Champanhe tower.

(c) Kendo.

(d) Clown.

(e) Dog.

Figure 3.7: 3D video sequences used to estimate the proposed video quality model.

compute the SSIM with reference to the corresponding uncorrupted decoded frames and the results stored (Figure 3.6 stage 5). The SSIM drop ($Delta\_SSIM$) is the difference between the SSIM of the error-free stereo frame (i.e., SSIM=1) and that of the displayed frame, assuming that frame-copy is used as the concealment method. Note that coding distortion is not included in this model, which means that SSIM=1 for any frame correctly received. The $Delta\_SSIM$ computation for each frame x is presented below, where the EF_SSIM_x is the error-free stereo frame SSIM and D_SSIM_x is the displayed frame SSIM.

$$Delta\_SSIM_x = EF\_SSIM_x - D\_SSIM_x \tag{3.1}$$

The $Delta\_SSIM$ is plotted in Figure 3.8 as a function of lost frame size for P and B slices using Dataset-2, and in Figure 3.9 using Dataset-3. It can be seen that the point cloud thickness is higher for B slices than P slices, this means that the error between the actual $Delta\_SSIM$ and the polinomial fitting is higher. A similar behavior was found from the experiments carried out with Datasets-1 used in this study. Linear, quadratic and cubic polynomials were then used as fitting models for the data obtained from the experiments with the three Datasets. The generic equation that defines the proposed models is the polynomial in Eq. 3.2 where $dSSIM_n$ is the $Delta\_SSIM$, n is the polynomial degree, L_{fs} is the lost frame size in bytes and p_n are the polynomial coefficients.

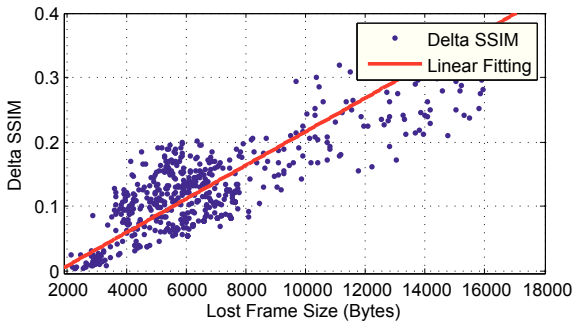
$$dSSIM_n = p_n L_{fs}^n + ... + p_2 L_{fs}^2 + p_1 L_{fs} + p_0 \tag{3.2}$$

The analysis of fitting models of different degrees concluded that third order models are accurate enough to estimate the dependency between lost frame size and $dSSIM_n$. The polynomial coefficients pertaining to each fitted curve are shown in Table 3.2 for P-frames and Table 3.3 for B-frames.

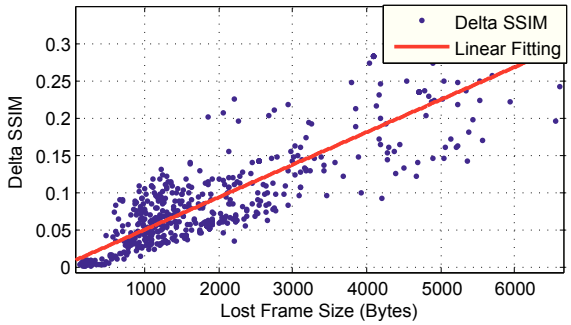
Linear fitting for P and B-frames is depicted in Figures 3.8(a) and 3.8(b), respectively. It can be seen the red linear fitting line crosses the middle of the $Delta\_SSIM$ points. As the $Delta\_SSIM$ get more dispersed, the fitting error increases. The actual $Delta\_SSIM$, thus a way to work around this is to use higher order fitting.

The second polinomial order fitting is illustrated in Figures 3.8(c) and 3.8(d), respectively. This quadratic fitting decreases the fitting error, since it follows the $Delta\_SSIM$ values shape.

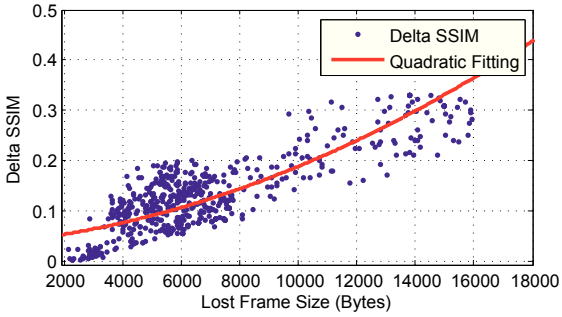
To improve the fitting error, cubic polinomial fitting was used. Figures 3.8(e) and 3.8(f) depicts the Dataset-2 cubic fitting for P and B-frames, respectively. Figure 3.9 illustrates the similar results for Dataset-3. As it can be seen, the cubic fitting better represents the $Delta\_SSIM$ values in function of the frame size.



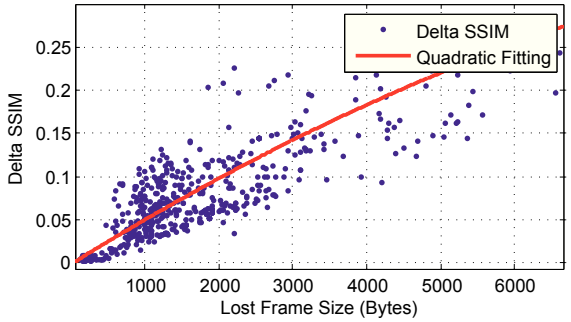
(a) Linear fitting for P slices.



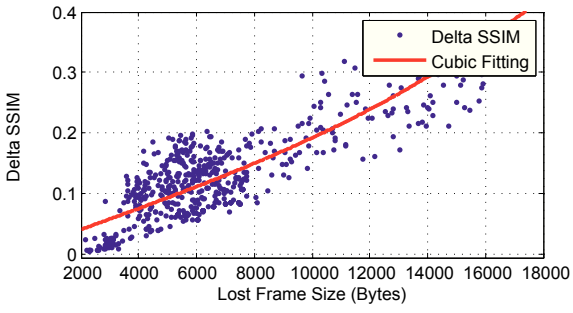
(b) Linear fitting for B slices.



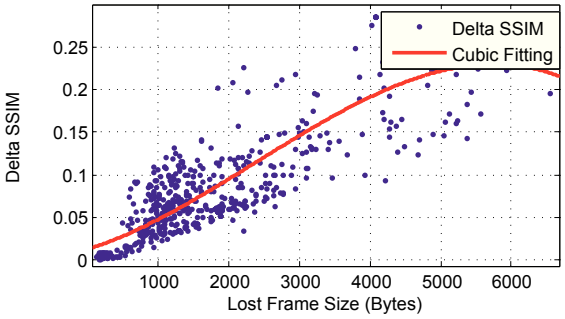
(c) Quadratic fitting for P slices.



(d) Quadratic fitting for B slices.

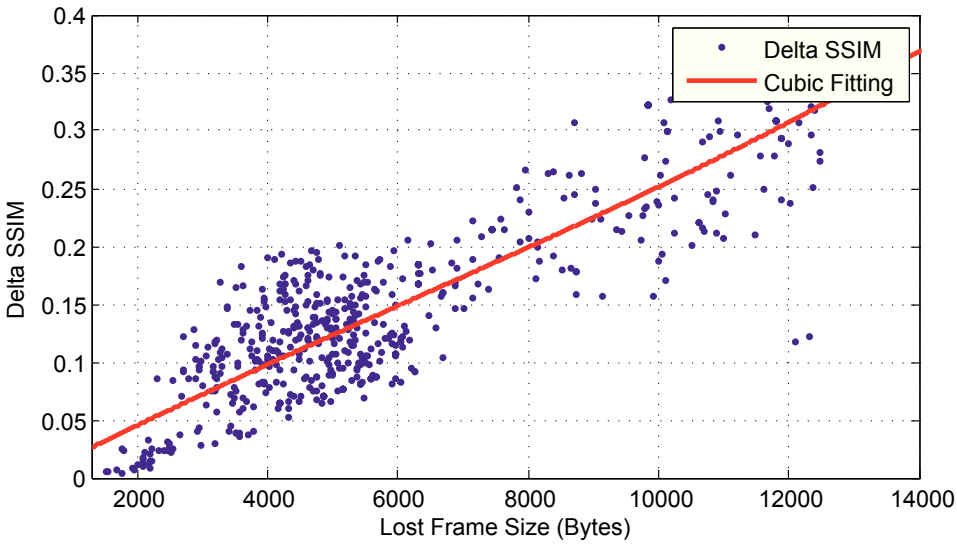


(e) Cubic fitting for P slices.

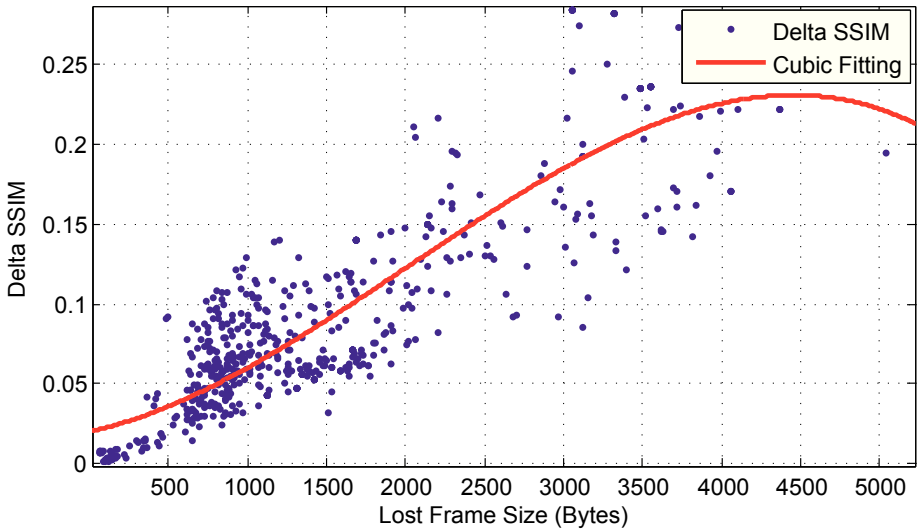


(f) Cubic fitting for B slices.

Figure 3.8: Delta SSIM vs frame size for Dataset-2.



(a) Cubic fitting for P slices.



(b) Cubic fitting for B slices.

Figure 3.9: Delta SSIM vs frame size for Dataset-3.

Table 3.2: Curve fitting coefficients for P-frames.

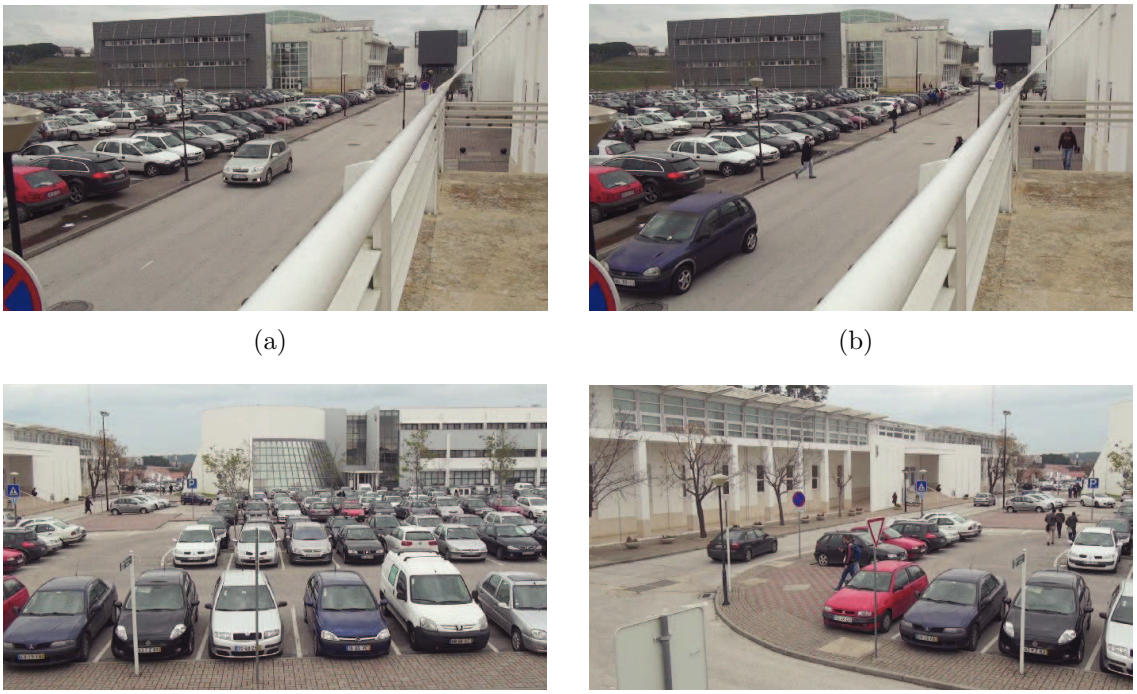
	Linear	Quadratic	Cubic
Dataset-1	p0=0.03596 p1=3.66E-06 - -	p0=0.175 p1=-2.11E-05 p2=9.26E-10 -	p0=0.05365 p1=9.29E-06 p2=-1.19E-09 p3=4.22E-14
Dataset-2	p0=-0.04488 p1=2.61E-05 - -	p0=3.89E-02 p1=6.11E-06 p2=8.92E-10 -	p0=4.74E-03 p1=1.78E-05 p2=-1.87E-10 p3=2.72E-14
Dataset-3	p0=-0.1276 p1=9.01E-05 - -	p0=-0.2097 p1=1.43E-04 p2=-6.66E-09 -	p0=-0.03292 p1=-2.92e-05 p2=3.86E-08 p3-3.28E-12

Table 3.3: Curve fitting coefficients for B-frames.

	Linear	Quadratic	Cubic
Dataset-1	p0=0.01636 p1=3.05E-05 - -	p0=-1.83E-02 p1=5.69E-05 p2=-3.48E-09 -	p0=-1.90E-02 p1=5.78E-05 p2=-3.77E-09 p3=2.57E-14
Dataset-2	p0=0.006689 p1=4.38E-05 - -	p0=-2.04E-03 p1=5.34E-05 p2=-1.80E-09 -	p0=1.30E-02 p1=2.57E-05 p2=1.07E-08 p3=-1.50E-12
Dataset-3	p0=-0.006671 p1=5.65E-05 - -	p0=2.07E-03 p1=6.29E-05 p2=-1.54E-09 -	p0=2.01E-02 p1=2.13E-05 p2=2.23E-08 p3=-3.69E-12

3.4 Simulation results

To validate the proposed models a 3D video sequence was captured to simulate a real-time case analysis. Towards that end the stereo video sequence was captured using a 3D Digital HD Video Camera Recorder (Sony HXR-NX3D1U NXCAM) with approximately 4000 stereoscopic frames. A busy hour in ESTG's parking car was the scenario chosen for the capturing, because it has several motion features (e.g., driving cars, walking people) and also a complex background scenario with all the parked cars and trees. Figure 3.10 depicts a sample of the captured frames for different point of views.



(a) (b)

(c) (d)

Figure 3.10: Different views of captured 3D video sequence in Escola Superior de Tecnologia e Gestão de Leiria.

To evaluate single frame quality degradation one frame was lost in each GOP. The following configurations were used for stereo video sequence encoding:

- **Frame resolution**: 1920x1080
- **Frame-rate**: 30Hz
- **GOP size**: 21 frames
- **GOP structure**: IBPBP
- **I-frame QP**: 38

- **P-frame QP**: 40
- **B-frame QP**: 40

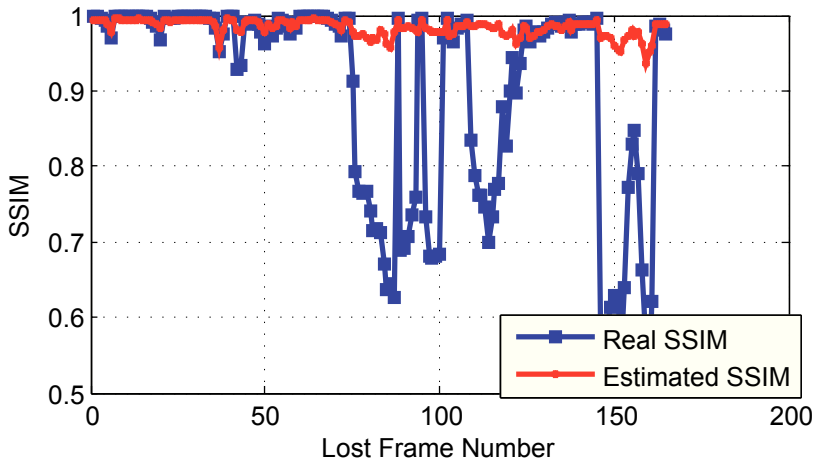
In Figure 3.11 and 3.12 the SSIM for the lost frames was estimated using linear, quadratic and cubic polynomial models and compared to the SSIM of the concealment frame that stands in for lost B and P-frame, respectively. It was found that the SSIMs estimated with the proposed models are quite similar to the real SSIM for P-frames, however for B-frame this is not that linear. In Figure 3.11 it can be seen that the shape of the real and estimated curve are quite similar, but with a scale coefficient that minimizes the estimated SSIM from the real one. This phenomenon is justified with RMSE and pearson correlation values presented in Table 3.4.

The model performance was then evaluated with two different performance indicators of the SSIM estimators: Root Mean Square Error (RMSE) and the Pearson Correlation values for both P and B frames and the three model orders. Their correlation measurements are presented in Table 3.4 using Dataset-3 coefficients to estimate the video quality. For P-frames, the RMSE decreases sightly with the increase of polynomial degree. As expected the Pearson Correlation increases with the polynomial degree. A similar behavior is observed for B-frames, though the model does not evidence such a strong correlation as that observed for the P-frames possibly due to the smaller size of B-frames and larger dispersion of the lost frame size vs. $dSSIM_n$ data around the fitted models.

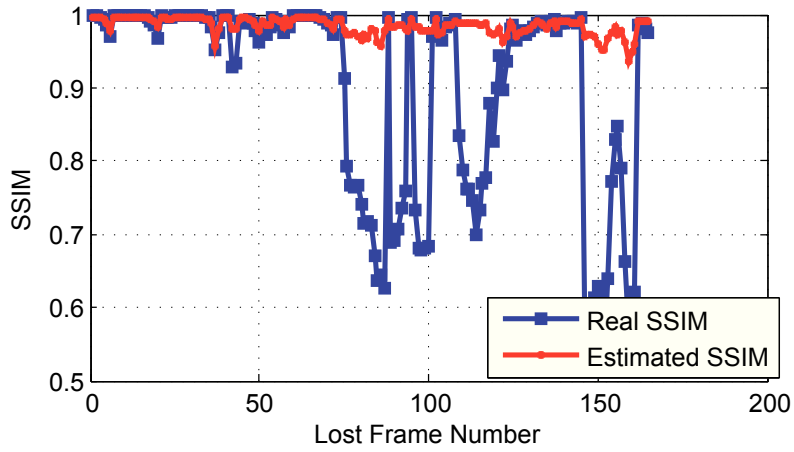
Table 3.4: Simulation results using Dataset-3 polinomial coefficients.

Polynomial		P-frame	B-frame
Linear	RMSE	0.1045	0.1556
	Pearson Corr.	0.783	0.7099
Quadratic	RMSE	0.0965	0.1562
	Pearson Corr.	0.8197	0.7106
Cubic	RMSE	0.0956	0.1655
	Pearson Corr.	0.8224	0.7073

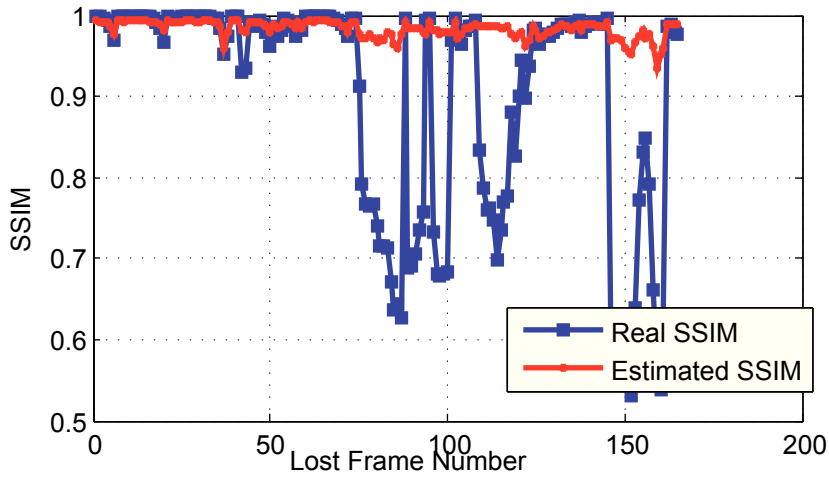
The simulation results show a strong correlation between the estimated and the actual 3D video quality. Both RMSE and Pearson Correlation measurements show very promising values if one considers that a quality measure in the visual domain is being estimated using information obtained from the compressed packetized bitstream domain, without accessing any reference information for comparison.



(a) Linear polynomial fitting.

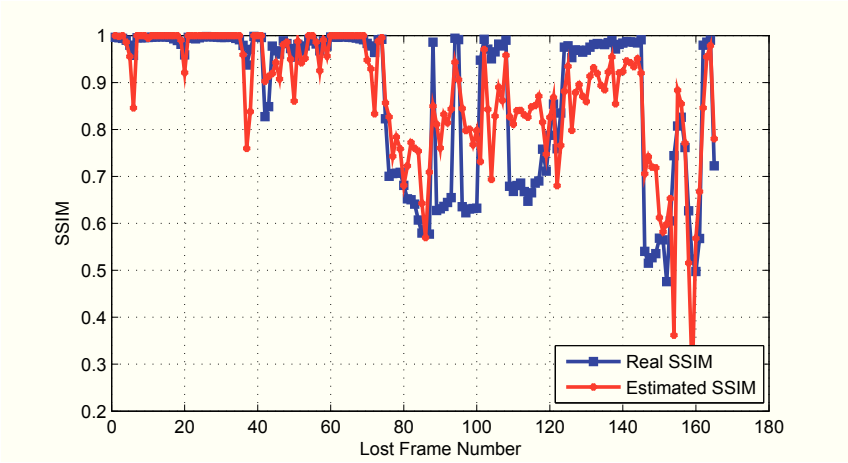


(b) Quadratic polynomial fitting.

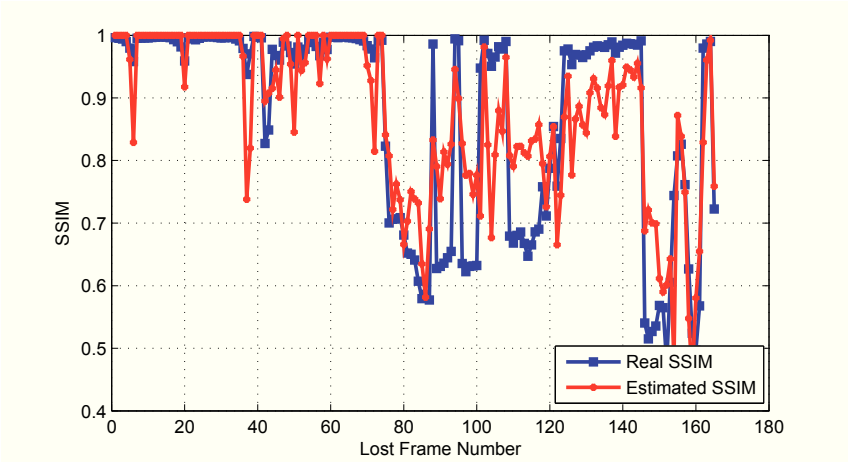


(c) Cubic polynomial fitting.

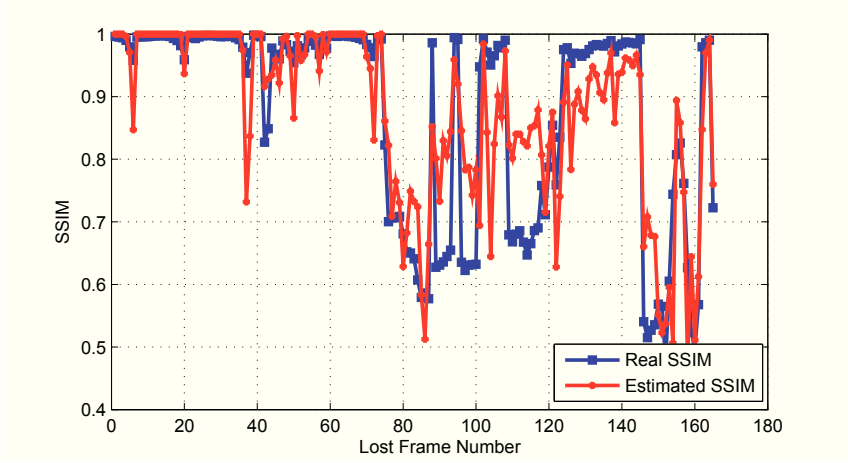
Figure 3.11: Estimated SSIM vs. real SSIM of lost B-Frames.



(a) Linear polynomial fitting.



(b) Quadratic polynomial fitting.



(c) Cubic polynomial fitting.

Figure 3.12: Estimated SSIM vs. real SSIM of lost P-Frames.

3.5 Summary

In this Chapter a method that estimates the objective quality measured as the SSIM is proposed, based on the size of the lost frames.

The results show that SSIM of missing stereoscopic frames in 3D coded video can be predicted with Root Mean Square Error (RMSE) accuracy of about 0.1 and Pearson correlation coefficient of 0.8, taking the SSIM of uncorrupted frames as reference. It is concluded that the proposed model is capable of estimating the SSIM quite accurately using only the estimated sizes of the lost frames.

Chapter 4

Software implementation

In this Chapter the application software is presented and its implementation is described. The Chapter is organised as follows. In a first stage an overview of the software is presented where the information flow and high-level functional analysis is described, along with the structure and implementation of the most important characteristics. The processing software responsible for the extraction of packet headers parameters and proposed mathematical model implementation is specified in Section 4.3. In Section 4.4 the Guide User Interface (GUI) is described, with particular emphasis on the input parameters, output display results and other features. Finally a summary of this Chapter is presented in Section 4.5.

4.1 Software overview

Considering the objective of the work (monitoring 3D video quality), it was necessary not only to devise a quality model, but also to develop an application software to implement the proposed quality model, and then fulfill the monitor requirements (i.e., mathematical model plus pratical implementation).

The monitor software implementation has two main processes, the video quality assessment thread and the Guide User Interface (GUI) thread. In Figure 4.1 one can see an overview of the interaction between the processes. The information flow from the GUI to monitor software (i.e., input parameters) and return flow from GUI of output display are represented, after the video quality assessment. This cycle will repeat itself until the monitor receives a command to stop or the end of video stream is reached.

The software was implemented in C++ language in Qt Creator. This IDE was chosen since the tools available to develop the user interface are considerably satisfatory. Regarding the menu bar and the real time graphics implementation, this IDE meets the

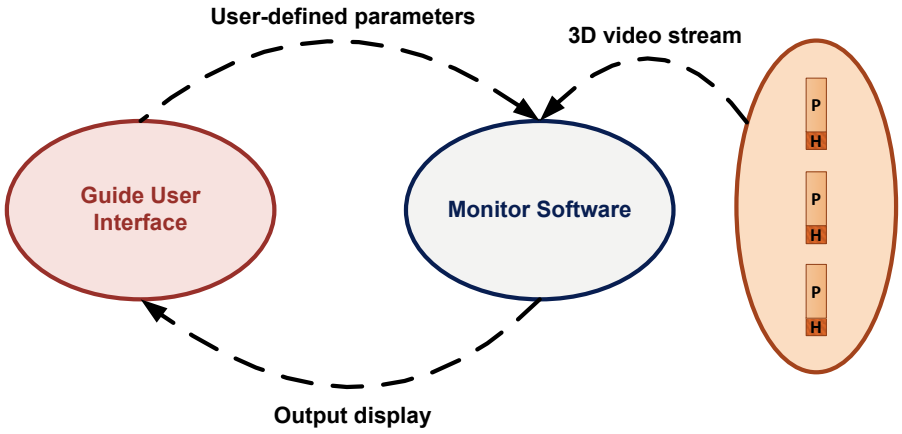


Figure 4.1: Software overview diagram.

requirements of the monitor implemented. The Objective-Oriented Programming (OOP) approach was used, due to the advantages of using objects and data fields (i.e., attributes that describe the object) and their associated procedures known as methods. This objects (i.e., instances of classes) are used to interact with design applications. These relations are described in Figure 4.2 where a class have several features known as attributes that describe the object and also have methods which define the behavior to be exhibited by instances of the associated class. The monitor software requires a user interface to input parameters and and output the results in graphical form. In the next section is explained in an high-level manner which classes were used and how they interact with each other.

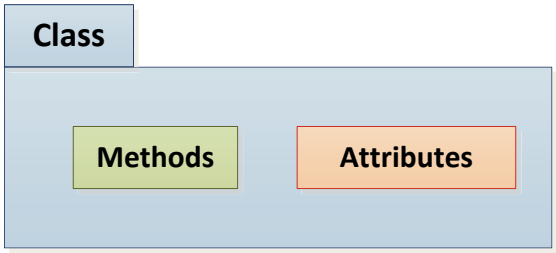


Figure 4.2: OOP classes features.

4.2 High-level design

The software was designed in several modules, where the classes are independent from each other. The classes that compose the monitor software are: monitor, access unit, ts, pes, nal, and frame. These names have been chosen to suggest their functions and features and they have been mentioned along this dissertation. As the name suggests

the class monitor has the main function of the software, which is monitoring the video quality. And thus, it is responsible to control the other classes functions and also the flow information between the GUI and the processing software. A set of NAL units in a specified form is referred to as an access unit. The decoding of each access unit results in one decoded picture. In H.264/stereo profile one access unit has two frames, one for each view. Thereafter, a class access unit was used to deal with the baseview and non-baseview frames. For that, it is needed to first parse the packet layers that compose the stereo video stream. For each packet layer a class was created (TS, PES, NAL), in each one is necessary to synchronise with the packet header start code that indicates the packet begin. And finally, extract the required parameters for the video quality evaluation. The class frame uses that information to stereo video quality assessment for each stereo-pair. The frame level information is also available in the class monitor, since it is the one responsible for the output information management. These dependencies between classes are illustrated in Figure 4.3, where one can see that an object of access unit or frame classes are instances of a monitor object. In turn, the TS, PES, NAL, and FRAME objects are instances of a access unit object.

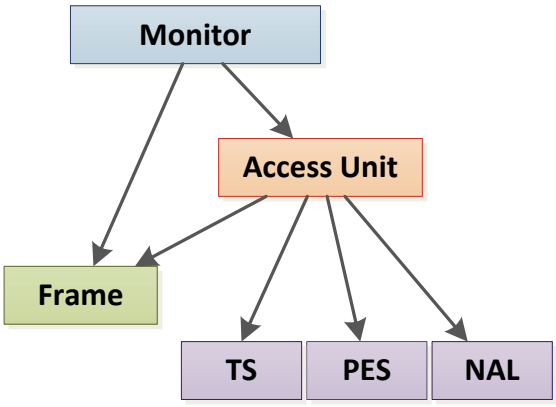


Figure 4.3: Classes dependencies diagram.

Each class has its procedures known as methods, and each method has its own actions and purpose. They exchange information between them according to their functions. Figure 4.4 illustrates the methods of each class and its interactions. It can be seen that the class monitor has a method called process that runs all the monitoring process. This process method controls another method ($syncTS$) of monitor class, the $syncTS$ method detects the start code that indicates the beginning of a TS packet, and makes sure that it is actually the starting point of the video stream. The access unit class methods controlled by the process method are described next:

- $readAU$: contains the procedures responsible for the stereo video stream parsing and the packet header parameters needed for quality assessment;

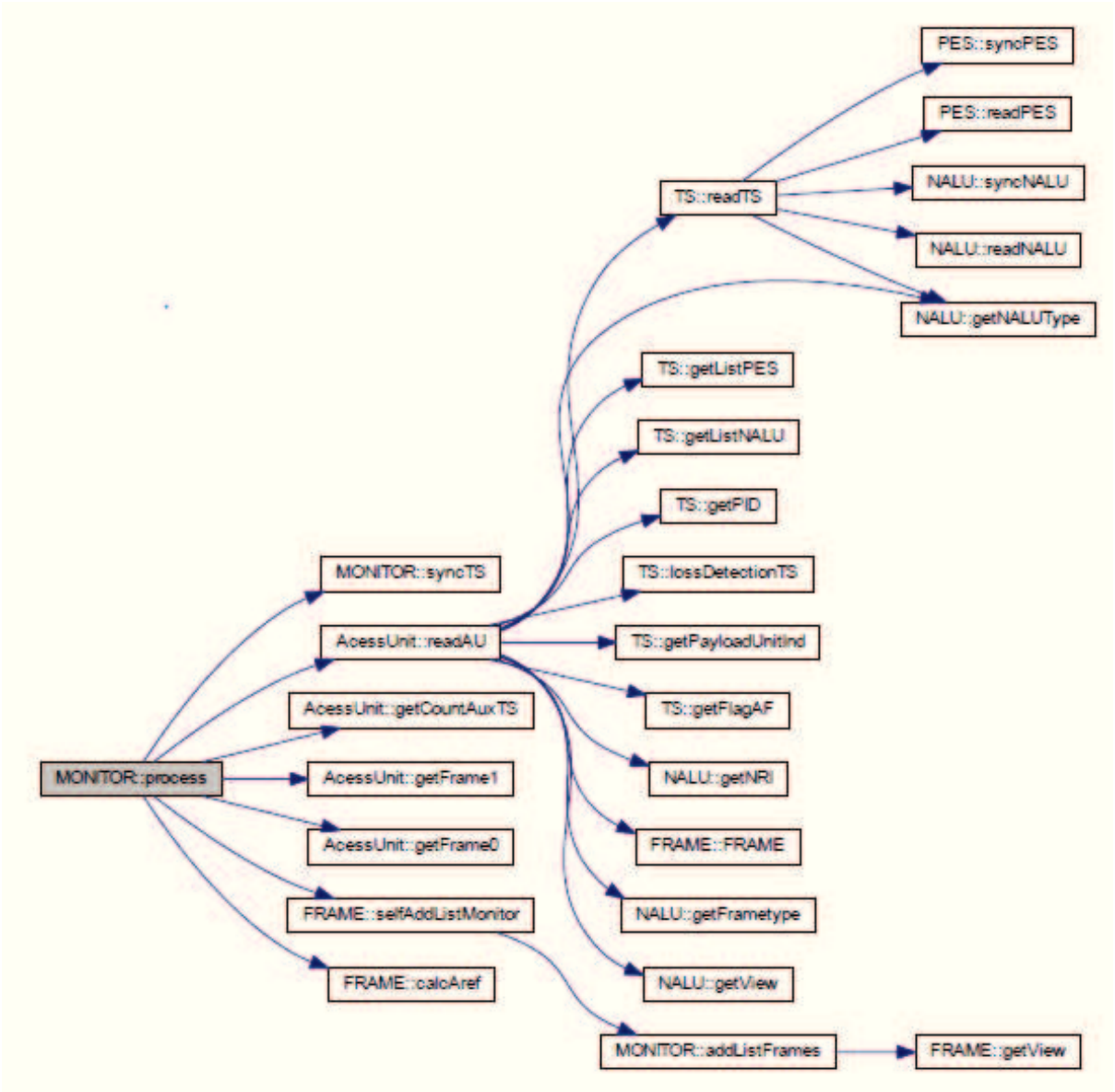


Figure 4.4: Classes methods diagram.

- *getCountAuxTS*: auxiliar TS packet counter, required to control the packet loss events;
- *getFrame1*: returns an instance of the frame class with non-basebview frame parameters (e.g., size, type, number, etc);
- *getFrame0*: returns an instance of the frame class with baseview frame parameters;

ReadAU method is the access unit main method, it controls the video stream parsing (i.e., *TS::readTS*), the TS packet loss detection (i.e., *TS::lossDetectionTS*) and all the "get" methods responsible for returning the needed information in the higher class levels (i.e., *TS::getListPES*, *TS::getListNALU*, *TS::getPID*, *TS::getPayloadUnitInd*, *TS::getFlagAF*, *NALU::getNRI* and *NALU::getView*) described next:

- *getListPES*: returns a list of PES class;
- *getListNALU*: returns a list of NAL class ;
- *getPID*: returns the value of packet ID;
- *getPayloadUnitInd*: returns the value of the Payload Unit Start Indicator, which indicates the start of the PES data is carried in the payload of the current TS packet;
- *getFlagAF*: returns the value of the TS packet header adaptation field, required to determine the size of the TS packet header and thus the paylod beginning;
- *getNRI*: returns the nal_ref_idc value, which indicates that the content of the NAL unit is not used to reconstruct reference pictures for inter picture prediction;
- *getView*: returns the view of the NAL unit;
- *getNALUType*: returns the NAL unit type (e.g., I, P or B);

ReadTS method controls the subroutines responsible for PES and NAL classes synchronization (i.e., *syncPES* and *syncNALU*) and packet header extraction (i.e., *readPES* and *readNALU*). Monitor class needs to know the frame class attributes, but since the frame class is an instance of access unit class which is an instance of monitor class, to solve this circular dependency was used. Circular dependency is a relation between two or more modules which either directly or indirectly depend on each other to function properly. Such modules are also known as mutually recursive. Thereby, the monitor class knows the frame class attributes, and thus is able to update the GUI with stereo video quality estimated in one of the frame class methods (i.e., *FRAME::calcAref*).

4.3 Processing software

The next step in the monitor's implementation is how the input information (i.e., video stream and users parameters) will be processed to compute the output display results (i.e., 3D video quality). This Section presents the processing software implementation and explain it step-by-step. The monitor's flowchart is depicted in Figure 4.5 where the software's routines and processes are labeled as well as the information flow between the GUI and processing software. The processing software's documentation is presented in Appendix B, where the classes, methods an attributes are detailed described. The information flow and processes descriptions are step-by-step presented bellow.

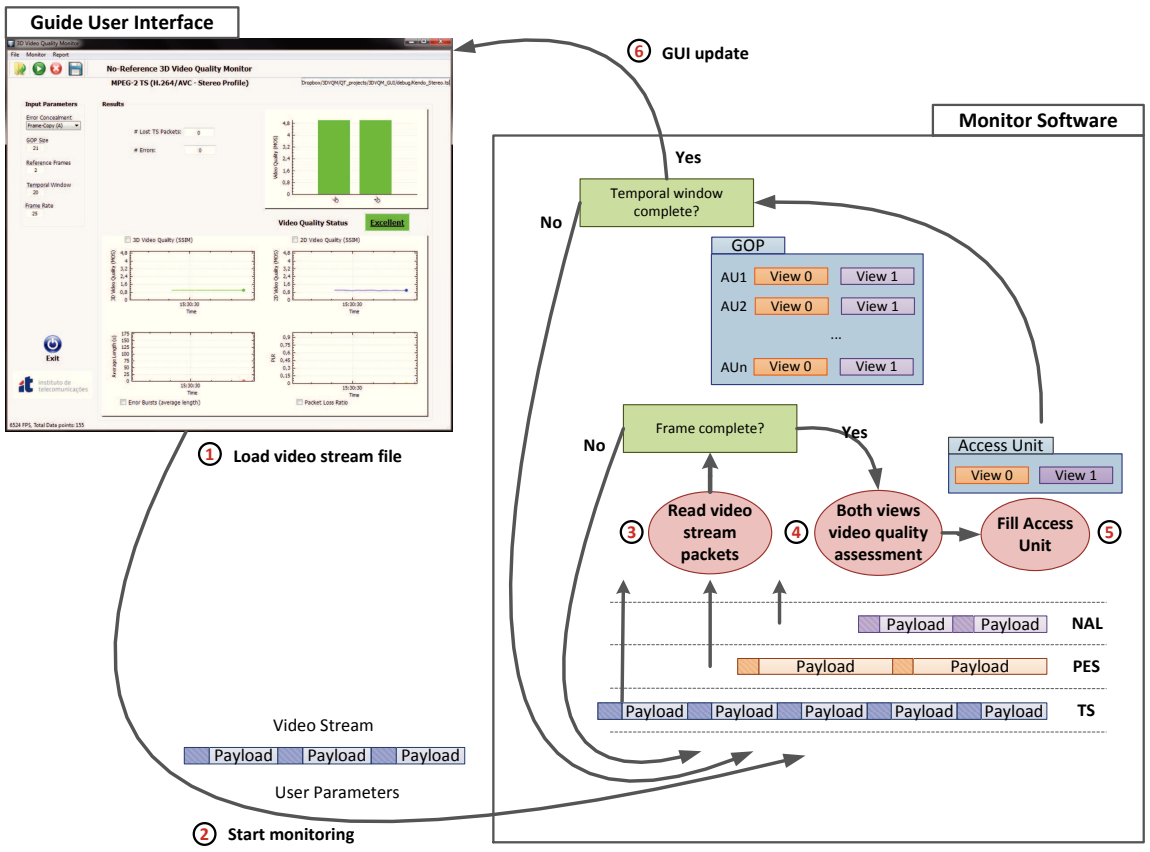


Figure 4.5: Monitor's flowchart.

- **1 - Load video stream file:** the monitor's first step is to load the 3D video information encapsulated in MPEG-2 TS stream file. The file path is then displayed in its text box;
- **2 - Start monitoring:** after loading the file, the software is able to start monitoring. The start button starts the software processing thread;

- **3 - Read video stream packets:** to access the video quality several parameters need to be extracted from the stream (e.g., packet headers information). This process has three sub-routines, one to read all the **TS packet headers** for each frame, the other to read the frame's **PES packet header** and another to read all the **NAL Unit headers** for each frame;
- **4 - Both views quality assessment:** this estimated process implements the proposed quality model. Quality assessment is frame-by-frame for both base and auxiliar-views;
- **5 - Fill Access Unit:** when a complete video frame is read from the stream, the respective quality is estimated. Then, to enhance the monitor performance, the ouput information is divided by frame and Access Unit. Also, for each frame the output parameters are classified into packet-loss information (e.g., number of errors and lost TS packets), quality information and frame specific information (e.g. view, type, size);
- **6 - GUI update:** when the temporal window (in frames) is complete, the software sends the output information to the display and updates the GUI. The GUI's thread will refresh the graphics and the text information. If the end of stream is not found or no order to stop is received, the software monitor will continue this process by repeating the previous steps;

The monitor implementation was specifically designed to keep both processing and GUI's software completely independent from each other. These processes change information between them, but they operate independently. This customization feature increases the range of potential application. Given a remote monitoring scenario where the user wants to access the monitor at any place and at any time, one of the possibilities is to incorporate the interface in an HTML page. This feature allows such a scenario because the processing software as well as its inputs and outputs fulfills the necessary requirements, thereafter any GUI that fulfill similar conditions is able to interoperate with this monitor software. In the next section, the GUI input and output parameters and other functionalities are described.

4.4 Guide user interface

The theoretical model implementation in a pratical scenario brings some problems to solve, where probably the most important one is how the user will interact with the model, how the input and output transactions will be made? This Section provides the answer to these questions.

An important feature of this monitor software is the output display, where the users can visually understand what happens in real-time in terms of the video stream quality. Thereby, Figure 4.6 depicts the GUI and it's input and output parameters as well as other tools. These features are presented and explained bellow.

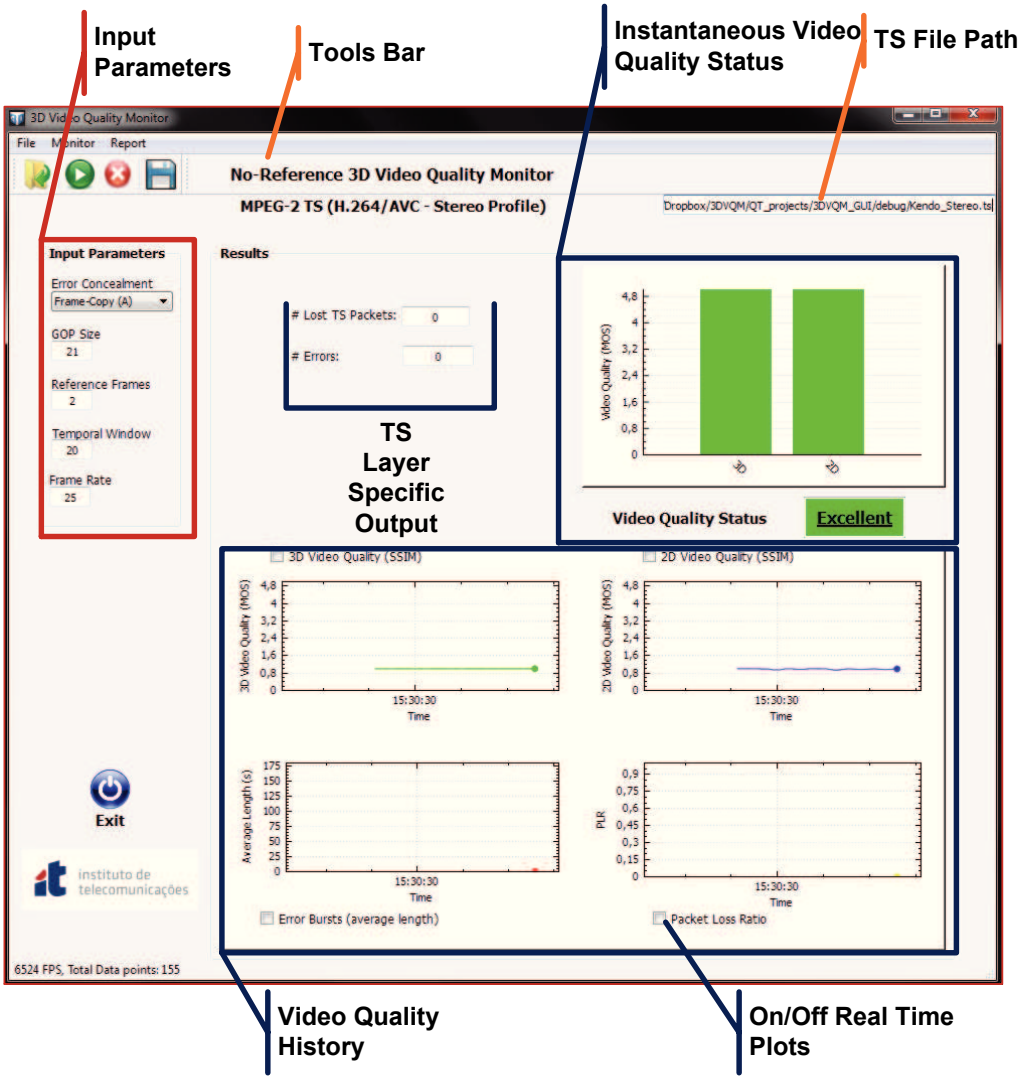


Figure 4.6: Guide user interface.

4.4.1 Input parameters

Beyond the corrupted video stream data input in monitor software, there are other input information that is needed for video stream parsing and quality assessment itself. In most pratical applications the video coding parameters are constant for each service provider,

and thus one can consider those parameters as fixed input parameters. In the monitor software these are classified as user-defined parameters as depicted in Figure 4.7 and described bellow.

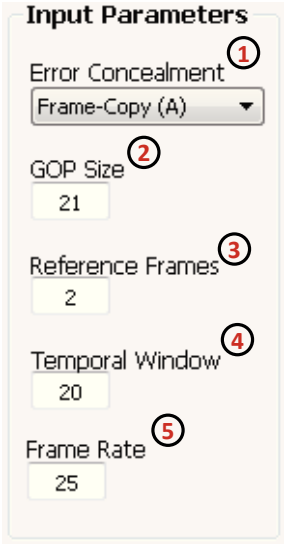


Figure 4.7: Input parameters.

- **1 - Error concealment type:** different techniques of error concealment are possible because they have different impact in video quality;
- **2 - GOP size:** necessary to identify the I-frames, and then understand where the propagation error ends;
- **3 - Reference frames:** number of frames depending of the current frame;
- **4 - Temporal window:** frequency of GUI refresh (i.e., how often the display results are updated), by default this is the same size of GOP;
- **5 - Frame rate:** frequency rate of displayed frames;

4.4.2 Output display

The quality assessment thread's output is displayed in the GUI in several different ways, such as text boxes, history graphics and instantaneous graphics. Figure 4.8 depicts de GUI's output display. It can be identified TS layer specific output, video quality history and instantaneous video quality status as also identified in Figure 4.6. Each output is labeled in 4.8 and described bellow.

Figure 4.8: Output display results.

- **1 - Lost TS packets:** number of lost TS packets in the video stream;
- **2 - Errors:** number of errors in the video stream;
- **3 - 3D video quality:** graphical history of 3D video quality (SSIM);
- **4 - Error bursts:** graphical history of errors bursts;
- **5 - Packet loss ratio:**graphical history of packet loss ratio;
- **6 - 2D video quality:** graphical history of 2D video quality (SSIM);
- **7 - Instantaneous video quality:** 2D and 3D instantaneous video quality (SSIM);
- **8 - Check buttons:** hide or show the corresponding graphics;

4.4.3 Other features

Another important GUI feature is the functionality and user friend characteristics enabled by Qt Creator IDE which is an excellent tool for this purpose. As it can be seen in Figure 4.9 it allows the customisation of the menu and tools bars, adding the desired icons as necessary. It also permits to add buttons and text boxes, as well as static images. All the extra functionalities are labeled in Figure 4.9 and described bellow.

Figure 4.9: Other features.

- **1 - Menu bar:** groups the functionalities by type;
- **2 - Open button:** opens a dialog box to select the video stream file and add it to the software;
- **3 - Start button:** after loading the file this button allows to start the application;

- **4 - Stop button:** allows to stop the monitoring process at any time;
- **5 - Save button:** after a monitoring period allows to save the quality information into a report file;
- **6 - Monitor description:** description of monitor type and video codec/profile used;
- **7 - Video stream file path:** path to load the video stream;
- **8 - Exit button:** allows to close the monitor and its processes at any time;
- **8 - Logo:** logo of the Telecommunications Institute;

At this stage is important to report not only the software's range of applications and functionalities, but also its limitations and weak points. During the implementation phase many obstacles were found. An important one was the GUI's freeze when the processing software was running. To solve this problem both processes were placed in different threads (i.e., on a multi-core system every processor or core executes a separate thread simultaneously). Thus, when the start button is turned on, the processing software will run it's sub-routines and all the GUI's functionalities are ready to use.

4.5 Summary

In this Chapter, the monitor software implementation was described, along with an overview in Section 4.1 that shows the high level monitor's implementation (e.g., how the user interface and processing software interact with each other). The monitor's and GUI's software deployment is described step-by-step in Sections 4.4 and 4.3, respectively.

This Chapter ends the description of the work done, after the literature review, the proposed model description, and its pratical implementation. In this case the overall result corresponds to the development of a monitor software that implements the proposed mathematical quality model, and then exhibits the output results for users analysis.

Chapter 5

Conclusion and future work

This chapter concludes this dissertation, presenting some conclusions about this research work and some future research perspectives in the field of 3D video quality monitoring.

5.1 Conclusions

This section discusses a summary of the chapters presented before and a review of the contributions presented.

Chapter 2 presented a review of the main NR video quality assessment models in the literature. Several Video Quality Assessment model types (e.g., packet-layer, bitstream level, media-layer and hybrid models) were presented. A study for 2D and 3D scenarios was described. Considering the lack of accurate models for 3D video and since 3D video can be seen as an extension of 2D video, the existing video quality assessment models can be expanded. The NR-VQA models under study were classified as packet-layer models, since the decoded video signal is not necessary to determine the quality of video, i.e., it only uses packet headers information. This was important to understand how those models work and which parameters/measurements they implement to the development of the proposed NR-VQA model.

Chapter 3 presented the mathematical model, as well as it's scenario and results. The scenario under study is a broadcast network (i.e., TS/PES/NAL), the error concealment method used was frame-copy. Coded distortion was not included in this model. A simulation study was undertaken with different 3D video streams. Frame sizes of lost frames were found to be correlated with the quality degradation incurred as a result of loss of those frames. It was concluded that a polynomial fitting is able to model this dependency. The model coefficients for different polynomial degrees were determined using three different datasets. The model was validated with a long real-life 3D video sequence. When

applied to this sequence (after introduction of artificial packet losses) the quality estimator performed quite well with Pearson Correlation coefficients around 0.8 which indicate a strong positive correlation, and RMSE values around 0.1 evidencing a small estimation error. These results are quite remarkable as the proposed packet layer model is based only on the estimated lost frame size, which contains very little information about the characteristics and visual content of the lost frame.

After the 2D and 3D VQA models review and the proposed model explanation, Chapter 4 presents the monitor's software implementation. The NR-VQA model proposed in 3 is deployed in the monitor's software. As inputs the software as several user-defined parameters and the corrupted video stream file and the quality results are displayed in the GUI for analysis in a user-friendly way. The results are presented in the form of temporal and instantaneous graphics. This monitor software closes the development of this work but several open issues for model and software improvements are identified in the next Section.

One may conclude that this research work was a relevant contribution and one IEEE international workshop publication (Appendix A). Nevertheless, the contribution of this is beyond the publications, since the model was accomplished in a pratical environment with the monitor's software implementation.

5.2 Future work

In this research work, a 3D video quality model was proposed and it's software implementation presented. Nevertheless, there are still a number of topics that require further investigation, and may lead to a higher model's accuracy and monitor's performance.

- **3D video quality extension to error propagation level.** The VQA model proposed in Chapter 3 determines stereo-pair frame quality degradation in a single-loss scenario. Since in a GOP, the stereo-pair frames depend from each other, the effects of temporal error propagation and propagation between views could increase the quality model's accuracy.

- **3D video quality extension to burst-loss scenario.** Another important upgrade is to take into account not only the single-loss frames, but also the burst-loss scenario where an adjacent set of information is lost, and thus has a different impact in the perceived video quality when compared with the single-loss scenario.

- **3DVQM software network implementation.** The software's monitor input is a TS video stream file. However in a real scenario the monitor should work at any

point of the packets network. This is a relevant add on to take into account, for an automatic and independent monitoring process.

- **Add monitor more network scenarios.** The scenario under study is a broadcast network (i.e., TS/PES/NAL). However, there are other network scenarios that transport video content, such as IPTV (e.g., UDP/RTP/NAL). It is important to make this monitor as much universal as possible to accommodate different protocol stacks.

As a summary, one may conclude that the 3D quality model presented brings new insights in objective NR VQA field. The results are quite promissing as the proposed packet layer model is based only on the estimated lost frame size, which contains very little information about the characteristics and visual content of the lost frame. The VQA model is implemented in a pratical scenario where a monitor application was developed to fulfill technical user requirements.

Bibliography

[1] H. R. Wu, K. R. Rao, and A. A. Kassim, "Digital video image quality and perceptual coding," *Journal of Electronic Imaging*, vol. 16, no. 3, p. 039901, 2007. [Online]. Available: http://link.aip.org/link/?JEI/16/039901/1

[2] G. Ghinea, P. Muntean, F. Etoh, F.and Speranza, and H. Wu, "Special issue on quality issues on mobile multimedia broadcasting," vol. 54, no. 3, 2008, pp. 424–727.

[3] A. S. Umar, R. M. Swash, and A. Sadka, "Subjective quality assessment of 3D videos," in *AFRICON, 2011*, 2011, pp. 1–6.

[4] V. Q. E. G. (VQEG), "Final report from the video quality expert group on the validation of objective models of video quality assessment, phase ii," in *Available: http://www.vqeg.org*, 2003.

[5] I. R. J.144, *Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference*, Mar. 2004.

[6] Z. Yu, H. R. Wu, S. Winkler, and T. Chen, "Vision-model-based impairment metric to evaluate blocking artifacts in digital video," *Proceedings of the IEEE*, vol. 90, no. 1, pp. 154 –169, jan 2002.

[7] P. Hao, Q. Shi, and Y. Chen, "Co-histogram and its application in remote sensing image compression evaluation," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3, sept. 2003, pp. III – 177–80 vol.2.

[8] J. Baina, P. Bretillon, D. Masse, and A. Refik, "Quality of MPEG2 signal on a simulated digital terrestrial television," *Broadcasting, IEEE Transactions on*, vol. 44, no. 4, pp. 381 –391, dec 1998.

[9] M. Carnec, P. Le Callet, and D. Barba, "An image quality assessment method based on perception of structural information," in *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, vol. 3, sept. 2003, pp. III – 185–8 vol.2.

[10] S. Olsson, M. Stroppiana, and J. Baina, "Objective methods for assessment of video quality : state of the art," *Broadcasting, IEEE Transactions on*, vol. 43, no. 4, pp. 487 –495, dec 1997.

[11] S. Argyropoulos, A. Raake, M.-N. Garcia, and P. List, "No-reference bit stream model for video quality assessment of H.264/AVC video based on packet loss visibility," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, May 2011, pp. 1169–1172.

[12] T. Yamada, Y. Miyamoto, and M. Serizawa, "No-reference video quality estimation based on error-concealment effectiveness," in *Packet Video 2007*, Nov. 2007, pp. 288 –293.

[13] J. Han, Y. han Kim, J. Jeong, and J. Shin, "Video quality estimation for packet loss based on no-reference method," in *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, vol. 1, Feb. 2010, pp. 418 –421.

[14] D. Sung, S. Hong, Y. Kim, Y. Kim, and T. P. J. Shin, "No reference quality assessment over packet video network," in *IWAIT 2009*, 2009.

[15] K.-H. Thung and P. Raveendran, "A survey of image quality measures," in *Technical Postgraduates (TECHPOS), 2009 International Conference for*, 2009, pp. 1–4.

[16] H. R. Wu, K. R. Rao, and A. A. Kassim, "Digital video image quality and perceptual coding." *J. Electronic Imaging*, vol. 16, no. 3, p. 039901, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/jei/jei16.html#WuRK07

[17] I. T. U. I. R. C. Sector:, "Methodology for the subjective assessment of the quality of television pictures," in *ITU-R BT. 500-11*, 2002.

[18] C. Sun, X. Liu, X. Xu, and W. Yang, "An efficient quality assessment metric for 3D video," in *Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on*, 2012, pp. 209–213.

[19] Z. W. Ligang, Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment using structural distortion measurement," in *in Proc. IEEE Int. Conf. Image Proc*, 2002, pp. 65–68.

[20] A. Takahashi, D. Hands, and V. Barriac, "Standardization activities in the ITU for a QoE assessment of IPTV," *Communications Magazine, IEEE*, vol. 46, no. 2, pp. 78–84, 2008.

[21] N. Liao and Z. Chen, "A packet-layer video quality assessment model with spatiotemporal complexity estimation," *EURASIP Journal on Image and Video Processing*, vol. 2011, no. 1, p. 5, 2011. [Online]. Available: http://jivp.eurasipjournals.com/content/2011/1/5

[22] A. Reibman, V. Vaishampayan, and Y. Sermadevi, "Quality monitoring of video over a packet network," *Multimedia, IEEE Transactions on*, vol. 6, no. 2, pp. 327 – 334, 2004.

[23] P. Perez, J. Macias, J. J. Ruiz, and N. Garcia, "Effect of packet loss in video quality of experience," *Bell Labs Technical Journal*, vol. 16, no. 1, pp. 91–104, 2011. [Online]. Available: http://dx.doi.org/10.1002/bltj.20488

[24] K. Yamagishi and T. Hayashi, "Parametric packet-layer model for monitoring video quality of IPTV services," in *Communications, 2008. ICC '08. IEEE International Conference on*, May 2008, pp. 110 –114.

[25] K. Yamagishi, K. Ushiki, T. Hayashi, and A. Takahashi, "Parametric packet-layer model for monitoring video quality of IPTV services," in *NTT Technical Review*, Apr. 2009, pp. 110 –114.

[26] J. Greengrass, J. Evans, and A. Begen, "Not all packets are equal, part 2: The impact of network packet loss on video quality," *Internet Computing, IEEE*, vol. 13, no. 2, pp. 74 –82, march-april 2009.

[27] K. Stuhlmuller, N. Farber, M. Link, and B. Girod, "Analysis of video transmission over lossy channels," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 6, pp. 1012 –1032, jun 2000.

[28] Y. Han, A. Men, K. Chang, and Z. Quan, "GOP-level transmission distortion modeling for video streaming over mobile networks," in *Information Assurance and Security, 2009. IAS '09. Fifth International Conference on*, vol. 1, Aug. 2009, pp. 91 –95.

[29] Y. Wang, Z. Wu, and J. Boyce, "Modeling of transmission-loss-induced distortion in decoded video," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 16, no. 6, pp. 716 – 732, 2006.

[30] Q. Huynh-Thu, P. Le Callet, and M. Barkowsky, "Video quality assessment: From 2D to 3D - challenges and future trends," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, 2010, pp. 4025 –4028.

[31] X. Xiang, D. Zhao, Q. Wang, S. Ma, and W. Gao, "Rate-distortion optimization wi th inter-view refreshment for stereoscopic video coding over error-prone networks," *Proc. SPIE*, vol. 7257, pp. 1 – 7, Jan. 2009.

[32] Y. Zhou, C. Hou, W. Xiang, and F. Wu, "Channel distortion modeling for multi-view video transmission over packet-switched networks," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 11, pp. 1679 –1692, Nov. 2011.

[33] S. Yasakethu, C. Hewage, W. Fernando, and A. Kondoz, "Quality analysis for 3D video using 2D video quality models," *Consumer Electronics, IEEE Transactions on*, vol. 54, no. 4, pp. 1969 –1976, Nov. 2008.

[34] A. Mittal, A. Moorthy, J. Ghosh, and A. Bovik, "Algorithmic assessment of 3D quality of experience for images and videos," in *Digital Signal Processing Workshop and IEEE Signal Processing Education Workshop (DSP/SPE), 2011 IEEE*, Jan. 2011, pp. 338 –343.

[35] L. Goldmann, F. De Simone, and T. Ebrahimi, "Impact of acquisition distortions on the quality of stereoscopic images," in *5th International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM)*, 2010.

[36] L. Goldmann, F. D. Simone, and T. Ebrahimi, "A comprehensive database and subjective evaluation methodology for quality of experience in stereoscopic video," A. M. Baskurt, Ed., vol. 7526, no. 1. SPIE, 2010, p. 75260S. [Online]. Available: http://link.aip.org/link/?PSI/7526/75260S/1

[37] C. Hewage, S. Worrall, S. Dogan, and A. Kondoz, "Prediction of stereoscopic video quality using objective quality models of 2-D video," *Electronics Letters*, vol. 44, no. 16, pp. 963 –965, 2008.

[38] K. Yamagishi, T. Kawano, and T. Hayashi, "Effect of difference in 2D video quality for left and right views on overall 3D video quality," in *Image Processing (ICIP), 2012 19th IEEE International Conference on*, 2012, pp. 605–608.

[39] C. Hewage and M. Martini, "Quality evaluation for real-time 3D video services," in *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, 2011, pp. 1–5.

[40] G.-H. Chen, C.-L. Yang, L.-M. Po, and S.-L. Xie, "Edge-based structural similarity for image quality assessment," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 2, 2006, pp. II–II.

[41] K. Ha and M. Kim, "A perceptual quality assessment metric using temporal complexity and disparity information for stereoscopic video," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, 2011, pp. 2525–2528.

[42] M. Solh and G. AlRegib, "A no-reference quality measure for DIBR-based 3D videos," in *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, 2011, pp. 1–6.

[43] C. Fehn, "Depth-Image-Based Rendering (DIBR), Compression and Transmission for a New Approach on 3D-TV," in *Proceedings of SPIE Stereoscopic Displays and Virtual Reality Systems XI*, 2004, pp. 93–104.

[44] A. Maalouf and M.-C. Larabi, "A no-reference color video quality metric based on a 3D multispectral wavelet transform," in *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*, 2010, pp. 11–16.

[45] K. Sohn and S. Ryu, "No-reference quality assessment for stereoscopic images based on binocular quality perception," pp. 1–1, 2013.

[46] A. Voronov, D. Vatolin, D. Sumin, V. Napadovskiy, and A. Borisov, "Towards automatic stereo-video quality assessment and detection of color and sharpness mismatch," in *3D Imaging (IC3D), 2012 International Conference on*, 2012, pp. 1–6.

[47] B. Lee and M. Kim, "No-reference PSNR estimation for HEVC encoded video," *Broadcasting, IEEE Transactions on*, vol. 59, no. 1, pp. 20–27, 2013.

[48] M.-J. Chen, L. Cormack, and A. Bovik, "No-reference quality assessment of natural stereopairs," *Image Processing, IEEE Transactions on*, vol. 22, no. 9, pp. 3379–3391, 2013.

[49] A. Maalouf and M.-C. Larabi, "Cyclop: A stereo color image quality assessment metric," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 2011, pp. 1161–1164.

[50] B. Julesz, *Foundations of Cyclopean Perception*. University of Chicago Press, 1971.

[51] *On Binocular Rivalry*. Institute for Perception RVO-TNO, National Defence Research Organization TNO, 1965. [Online]. Available: http://books.google.pt/books?id=FbiIHAAACAAJ

[52] T. Schierl and S. Narasimhan, "Transport and storage systems for 3-D video using MPEG-2 systems, RTP, and ISO file format," *Proceedings of the IEEE*, vol. 99, no. 4, pp. 671–683, 2011.

[53] I. R. G.1050, *Network model for evaluating multimedia transmission performance over Internet Protocol*, Mar. 2011.

Appendix A

Published papers

- B. Feitor, P. Assunção, J. Soares, L. Cruz, R. Marinheiro, "Objective quality prediction model for lost frames in 3D video over TS", in *Communications Workshops (ICC), 2013 IEEE International Conference on*, 2013, pp. 622–625.
- B. Feitor, P. Assunção, J. Soares, L. Cruz, R. Marinheiro, "No-Reference Quality Models for Single Frame Loss in 3D Video", in *ConfTele, 2013 Castelo Branco, Portugal*, may. 2013, pp. 1–4.

Appendix B

Software documentation

3D Video Quality Monitor

Software Documentation

Nov 2013

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 AA Class Reference

`#include <AA.h>`

Public Member Functions

- AA (int icodingOrder, int iviewOrder, float iwEC, float isigma0, float isigma1, float ib, int inframesRef)
- void readAU (FILE *f, char *p1, char *p2, char *p3)

Public Attributes

- list< TS > listTS
- char buffTS [188]

4.1.1 Detailed Description

Definition at line 11 of file AA.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AA::AA (int *icodingOrder*, int *iviewOrder*, float *iwEC*, float *isigma0*, float *isigma1*, float *ib*, int *inframesRef*) [inline]

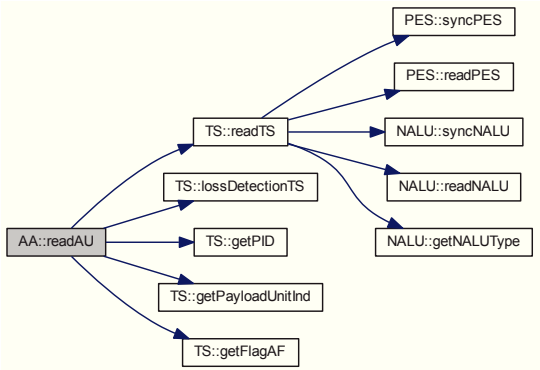
Definition at line 25 of file AA.h.

4.1.3 Member Function Documentation

4.1.3.1 void AA::readAU (FILE * *f*, char * *p1*, char * *p2*, char * *p3*)

Definition at line 21 of file AA.cpp.

Here is the call graph for this function:



4.1.4 Member Data Documentation

4.1.4.1 char AA::buffTS[188]

Definition at line 28 of file AA.h.

4.1.4.2 list<TS> AA::listTS

Definition at line 27 of file AA.h.

The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/AA.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/AA.cpp

4.2 AcessUnit Class Reference

`#include <AU.h>`

Public Member Functions

- AcessUnit (int codingOrder, int viewOrder, float wEC, float sigma0, float sigma1, int nframesRef, int countAux)
- int getCountAuxTS ()
- FRAME getFrame0 ()
- FRAME getFrame1 ()
- int readAU (FILE *f, char *p1, char *p2, char *p3)

Public Attributes

- list< TS > listTS
- char buffTS [188]

4.2.1 Detailed Description

Definition at line 17 of file AU.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 AcessUnit::AcessUnit (int *codingOrder*, int *viewOrder*, float *wEC*, float *sigma0*, float *sigma1*, int *nframesRef*, int *countAux*) [inline]

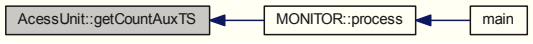
Definition at line 38 of file AU.h.

4.2.3 Member Function Documentation

4.2.3.1 int AcessUnit::getCountAuxTS ( ) [inline]

Definition at line 52 of file AU.h.

Here is the caller graph for this function:



4.2.3.2 FRAME AcessUnit::getFrame0 ( ) [inline]

Definition at line 56 of file AU.h.

Here is the caller graph for this function:

4.2.3.3 FRAME AcessUnit::getFrame1 ( ) [inline]

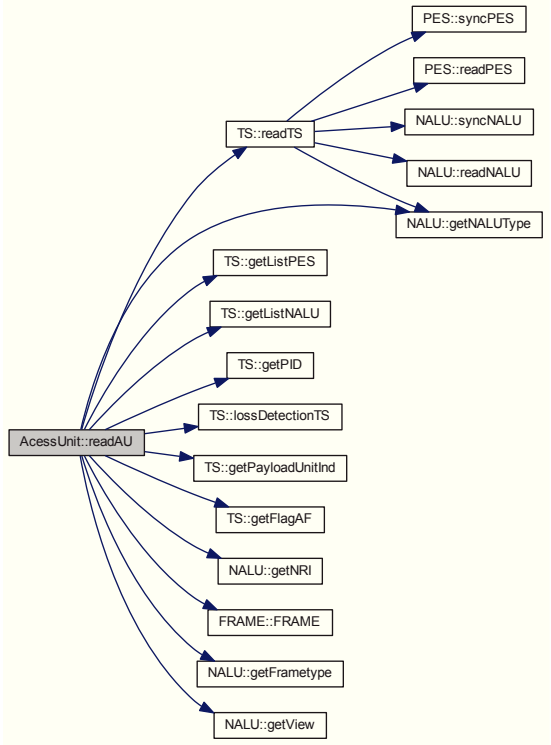
Definition at line 60 of file AU.h.

Here is the caller graph for this function:

4.2.3.4 int AcessUnit::readAU (FILE * *f*, char * *p1*, char * *p2*, char * *p3*)

Definition at line 21 of file AU.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

4.2.4 Member Data Documentation

4.2.4.1 char AcessUnit::buffTS[188]

Definition at line 64 of file AU.h.

4.2.4.2 list<TS> AcessUnit::listTS

Definition at line 49 of file AU.h.

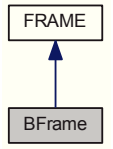
The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/AU.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/AU.cpp

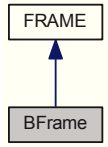
4.3 BFrame Class Reference

`#include <BFrame.h>`

Inheritance diagram for BFrame:



Collaboration diagram for BFrame:



Public Member Functions

- void selfAddListMonitor (MONITOR *monitor)
- void calcAref (vector< FRAME > *listFrames, float b, int nFrameRef)

4.3.1 Detailed Description

Definition at line 9 of file BFrame.h.

4.3.2 Member Function Documentation

4.3.2.1 void BFrame::calcAref (vector< FRAME > * *listFrames*, float *b*, int *nFrameRef*) [virtual]

Reimplemented from FRAME.

4.3.2.2 void BFrame::selfAddListMonitor (MONITOR * *monitor*) [virtual]

Reimplemented from FRAME.

Definition at line 20 of file BFrame.cpp.

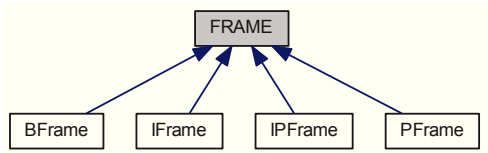
The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/BFrame.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/BFrame.cpp

4.4 FRAME Class Reference

`#include <FRAME.h>`

Inheritance diagram for FRAME:



Public Member Functions

- FRAME (char type, int size, int view, int packNum, int flagPacketLoss, int nPackLoss, int packlenAv, float wLoc, float wEP, float wEC)
- FRAME ()
- char getType ()
- int getSize ()
- int getView ()
- int getPackNum ()
- int getNPackLoss ()
- int getPackLenAV ()
- float getWLOC ()
- float getAref ()
- float getWEP ()
- float getWEC ()
- int getFlagPacketLoss ()
- float getD ()
- float getDi ()
- float getDpi ()
- float getDpe ()
- void setD (float d)
- void setDpi (float dpi)
- void setDi (float di)
- void setAref (float aref)
- void setDpe (float dpe)
- virtual void selfAddListMonitor (MONITOR *monitor)
- virtual void calcD ()
- void calcDi (float sigma0, float sigma1)
- void calcDpi (float sigma0, float sigma1, int nFramesRef, float b)
- virtual void calcAref (vector< FRAME > *listFrames, float b, int nFrameRef)

4.4.1 Detailed Description

Definition at line 9 of file FRAME.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 FRAME::FRAME (char *type*, int *size*, int *view*, int *packNum*, int *flagPacketLoss*, int *nPackLoss*, int *packlenAv*, float *wLoc*, float *wEP*, float *wEC*) [inline]

Definition at line 30 of file FRAME.h.

4.4.2.2 FRAME::FRAME ( ) [inline]

Definition at line 47 of file FRAME.h.

Here is the caller graph for this function:

4.4.3 Member Function Documentation

4.4.3.1 void FRAME::calcAref (vector< FRAME > *listFrames*, float *b*, int *nFrameRef*) [virtual]

Reimplemented in BFrame, IFrame, and PFrame.

Definition at line 4 of file BFrame.cpp.

Here is the caller graph for this function:

4.4.3.2 void FRAME::calcD ( ) [virtual]

Definition at line 17 of file IFrame.cpp.

4.4.3.3 void FRAME::calcDi (float *sigma0*, float *sigma1*)

Definition at line 38 of file FRAME.cpp.

4.4.3.4 void FRAME::calcDpi (float *sigma0*, float *sigma1*, int *nFramesRef*, float *b*)

Definition at line 49 of file FRAME.cpp.

4.4.3.5 float FRAME::getAref ( ) [inline]

Definition at line 77 of file FRAME.h.

4.4.3.6 float FRAME::getD ( ) [inline]

Definition at line 93 of file FRAME.h.

4.4.3.7 float FRAME::getDi ( ) [inline]

Definition at line 97 of file FRAME.h.

4.4.3.8 float FRAME::getDpe ( ) [inline]

Definition at line 105 of file FRAME.h.

4.4.3.9 float FRAME::getDpi ( ) [inline]

Definition at line 101 of file FRAME.h.

4.4.3.10 int FRAME::getFlagPacketLoss ( ) [inline]

Definition at line 89 of file FRAME.h.

4.4.3.11 int FRAME::getNPackLoss ( ) [inline]

Definition at line 65 of file FRAME.h.

4.4.3.12 int FRAME::getPackLenAV ( ) [inline]

Definition at line 69 of file FRAME.h.

4.4.3.13 int FRAME::getPackNum ( ) [inline]

Definition at line 61 of file FRAME.h.

4.4.3.14 int FRAME::getSize ( ) [inline]

Definition at line 53 of file FRAME.h.

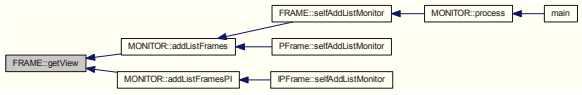
4.4.3.15 char FRAME::getType ( ) [inline]

Definition at line 50 of file FRAME.h.

4.4.3.16 int FRAME::getView ( ) [inline]

Definition at line 57 of file FRAME.h.

Here is the caller graph for this function:



4.4.3.17 float FRAME::getWEC ( ) [inline]

Definition at line 85 of file FRAME.h.

4.4.3.18 float FRAME::getWEP ( ) [inline]

Definition at line 81 of file FRAME.h.

4.4.3.19 float FRAME::getWLOC ( ) [inline]

Definition at line 73 of file FRAME.h.

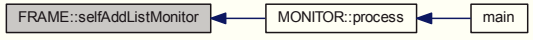
4.4.3.20 void FRAME::selfAddListMonitor (MONITOR * *monitor*) [virtual]

Reimplemented in BFrame, IFrame, PFrame, and IPFrame.

Definition at line 5 of file IFrame.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



4.4.3.21 void FRAME::setAref (float *aref*) [inline]

Definition at line 115 of file FRAME.h.

4.4.3.22 void FRAME::setD (float *d*) [inline]

Definition at line 109 of file FRAME.h.

4.4.3.23 void FRAME::setDi (float *di*) [inline]

Definition at line 113 of file FRAME.h.

4.4.3.24 void FRAME::setDpe (float *dpe*) [inline]

Definition at line 117 of file FRAME.h.

4.4.3.25 void FRAME::setDpi (float *dpi*) [inline]

Definition at line 111 of file FRAME.h.

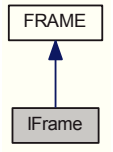
The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/FRAME.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/BFrame.cpp
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/FRAME.cpp
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/IFrame.cpp

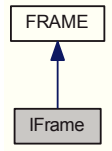
4.5 IFrame Class Reference

`#include <IFrame.h>`

Inheritance diagram for IFrame:



Collaboration diagram for IFrame:



Public Member Functions

- void selfAddListMonitor (MONITOR ∗monitor)
- void calcAref (vector< FRAME > ∗listFrames, float b, int nFrameRef)

4.5.1 Detailed Description

Definition at line 10 of file IFrame.h.

4.5.2 Member Function Documentation

4.5.2.1 void IFrame::calcAref (vector< FRAME > ∗ *listFrames*, float *b*, int *nFrameRef*) [virtual]

Reimplemented from FRAME.

4.5.2.2 void IFrame::selfAddListMonitor (MONITOR ∗ *monitor*) [virtual]

Reimplemented from FRAME.

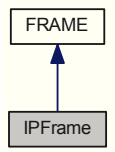
The documentation for this class was generated from the following file:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/IFrame.h

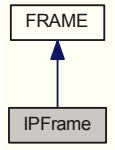
4.6 IPFrame Class Reference

`#include <IPFrame.h>`

Inheritance diagram for IPFrame:



Collaboration diagram for IPFrame:



Public Member Functions

- void selfAddListMonitor (MONITOR ∗monitor)

4.6.1 Detailed Description

Definition at line 11 of file IPFrame.h.

4.6.2 Member Function Documentation

4.6.2.1 void IPFrame::selfAddListMonitor (MONITOR ∗ *monitor*) [virtual]

Reimplemented from FRAME.

Definition at line 5 of file IPFrame.cpp.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/IPFrame.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/IPFrame.cpp

4.7 MONITOR Class Reference

`#include <MONITOR.h>`

Public Member Functions

- MONITOR ()
- MONITOR (float wEC, int GOPstructure, int nFrameRef, int Mframes)
- float getWEC ()
- int getMFrames ()
- int getGOPstructure ()
- int getNFrameRef ()
- float getB ()
- float getSigma0 ()
- float getSigma1 ()
- int getH ()
- int getV ()
- int openStream (int argc, char ∗∗argv)
- void process ()
- FILE ∗ syncTS (FILE ∗f)
- void addListFrames (FRAME ∗frame)
- void addListFramesPI (FRAME ∗frame)

Public Attributes

- FILE ∗ f

4.7.1 Detailed Description

Class that runs the processing routines for quality monitoring.

Version

1.0

Author

Bruno Feitor

4.7.2 Constructor & Destructor Documentation

4.7.2.1 MONITOR::MONITOR ( ) [inline]

Definition at line 49 of file MONITOR.h.

4.7.2.2 MONITOR::MONITOR (float *wEC*, int *GOPstructure*, int *nFrameRef*, int *Mframes*) [inline]

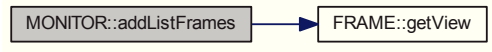
Definition at line 51 of file MONITOR.h.

4.7.3 Member Function Documentation

4.7.3.1 void MONITOR::addListFrames (FRAME * *frame*)

Definition at line 80 of file MONITOR.cpp.

Here is the call graph for this function:

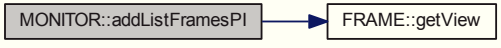


Here is the caller graph for this function:

4.7.3.2 void MONITOR::addListFramesPI (FRAME * *frame*)

Definition at line 88 of file MONITOR.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

4.7.3.3 float MONITOR::getB ( ) [inline]

Definition at line 78 of file MONITOR.h.

4.7.3.4 int MONITOR::getGOPstructure ( ) [inline]

Definition at line 70 of file MONITOR.h.

4.7.3.5 int MONITOR::getH ( ) [inline]

Definition at line 90 of file MONITOR.h.

4.7.3.6 int MONITOR::getMFrames ( ) [inline]

Definition at line 66 of file MONITOR.h.

4.7.3.7 int MONITOR::getNFrameRef ( ) [inline]

Definition at line 74 of file MONITOR.h.

4.7.3.8 float MONITOR::getSigma0 ( ) [inline]

Definition at line 82 of file MONITOR.h.

4.7.3.9 float MONITOR::getSigma1 ( ) [inline]

Definition at line 86 of file MONITOR.h.

4.7.3.10 int MONITOR::getV ( ) [inline]

Definition at line 94 of file MONITOR.h.

4.7.3.11 float MONITOR::getWEC ( ) [inline]

Definition at line 62 of file MONITOR.h.

4.7.3.12 int MONITOR::openStream (int *argc*, char ** *argv*)

Parameters

argc	string vector that includes all the input parameters to send to aplication.
argv	string vector that includes all the input parameters to send to aplication.

Returns

exception

Definition at line 10 of file MONITOR.cpp.

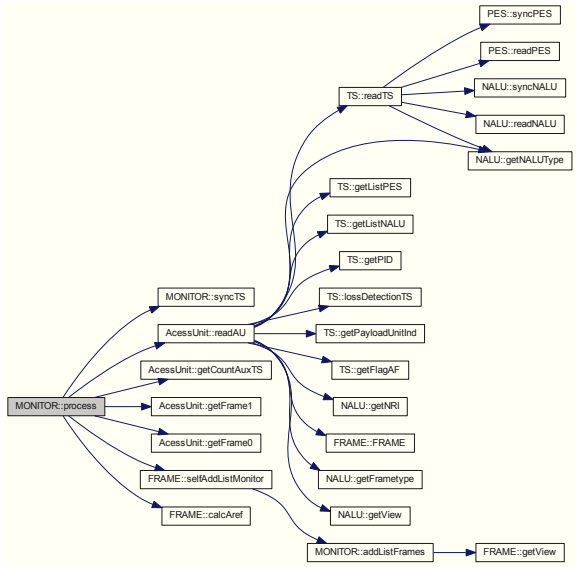
Here is the caller graph for this function:

4.7.3.13 void MONITOR::process ( )

This is the processing software's main routine.

Definition at line 27 of file MONITOR.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

4.7.3.14 FILE * MONITOR::syncTS (FILE * *f*)

This method stands for the synchronization with the TS stream packets.

Definition at line 96 of file MONITOR.cpp.

Here is the caller graph for this function:

4.7.4 Member Data Documentation

4.7.4.1 FILE* MONITOR::f

Definition at line 60 of file MONITOR.h.

The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/MONITOR.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/MONITOR.cpp

4.8 MonitorListener Class Reference

`#include <MonitorListener.h>`

Public Member Functions

- MonitorListener ()
- virtual void refreshGUI ()=0

4.8.1 Detailed Description

Definition at line 6 of file MonitorListener.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 MonitorListener::MonitorListener ( ) [inline]

Definition at line 13 of file MonitorListener.h.

4.8.3 Member Function Documentation

4.8.3.1 virtual void MonitorListener::refreshGUI ( ) [pure virtual]

The documentation for this class was generated from the following file:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/MonitorListener.h

4.9 NALU Class Reference

`#include <NALU.h>`

Public Member Functions

- NALU ()
- int getNALUType ()
- int getView ()
- int getNRI ()
- int getLength ()
- char getFrametype ()
- int syncNALU (char *pbuff, int endPES)
- void readNALU (char *pbuff, int beginNALU)

4.9.1 Detailed Description

Definition at line 21 of file NALU.h.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 NALU::NALU ( ) [inline]

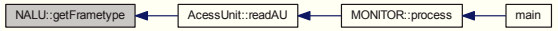
Definition at line 35 of file NALU.h.

4.9.3 Member Function Documentation

4.9.3.1 char NALU::getFrametype ( ) [inline]

Definition at line 54 of file NALU.h.

Here is the caller graph for this function:



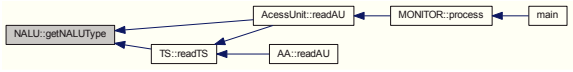
4.9.3.2 int NALU::getLength ( ) [inline]

Definition at line 50 of file NALU.h.

4.9.3.3 int NALU::getNALUType ( ) [inline]

Definition at line 38 of file NALU.h.

Here is the caller graph for this function:



4.9.3.4 int NALU::getNRI ( ) [inline]

Definition at line 46 of file NALU.h.

Here is the caller graph for this function:

4.9.3.5 int NALU::getView ( ) [inline]

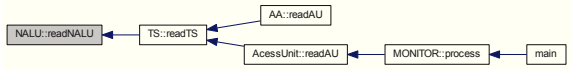
Definition at line 42 of file NALU.h.

Here is the caller graph for this function:

4.9.3.6 void NALU::readNALU (char * *pbuff*, int *beginNALU*)

Definition at line 25 of file NALU.cpp.

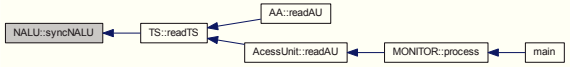
Here is the caller graph for this function:



4.9.3.7 int NALU::syncNALU (char * *pbuff*, int *endPES*)

Definition at line 5 of file NALU.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/NALU.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/NALU.cpp

4.10 PES Class Reference

`#include <PES.h>`

Public Member Functions

- PES ()
- int getLength ()
- int getStreamID ()
- int getSizeAUX ()
- int syncPES (char *pbuff, int sizeAF)
- int readPES (char *pbuff, int beginPES)

4.10.1 Detailed Description

Definition at line 6 of file PES.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 PES::PES ( ) [inline]

Definition at line 17 of file PES.h.

4.10.3 Member Function Documentation

4.10.3.1 int PES::getLength ( ) [inline]

Definition at line 19 of file PES.h.

4.10.3.2 int PES::getSizeAUX ( ) [inline]

Definition at line 27 of file PES.h.

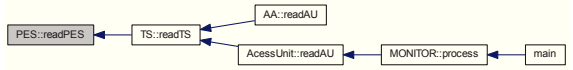
4.10.3.3 int PES::getStreamID ( ) [inline]

Definition at line 23 of file PES.h.

4.10.3.4 int PES::readPES (char * *pbuff*, int *beginPES*)

Definition at line 40 of file PES.cpp.

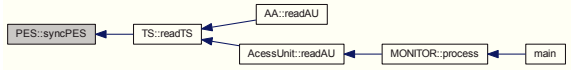
Here is the caller graph for this function:



4.10.3.5 int PES::syncPES (char * *pbuff*, int *sizeAF*)

Definition at line 18 of file PES.cpp.

Here is the caller graph for this function:



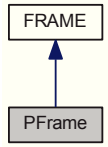
The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/PES.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/PES.cpp

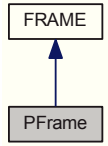
4.11 PFrame Class Reference

`#include <PFrame.h>`

Inheritance diagram for PFrame:



Collaboration diagram for PFrame:



Public Member Functions

- void selfAddListMonitor (MONITOR *monitor)
- void calcAref (vector< FRAME > *listFrames, float b, int nFrameRef)

4.11.1 Detailed Description

Definition at line 8 of file PFrame.h.

4.11.2 Member Function Documentation

4.11.2.1 void PFrame::calcAref (vector< FRAME > * *listFrames*, float *b*, int *nFrameRef*) [virtual]

Reimplemented from FRAME.

4.11.2.2 void PFrame::selfAddListMonitor (MONITOR * *monitor*) [virtual]

Reimplemented from FRAME.

Definition at line 5 of file PFrame.cpp.

Here is the call graph for this function:

The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/PFrame.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/PFrame.cpp

4.12 REPORT Class Reference

`#include <REPORT.h>`

Public Member Functions

- REPORT ()
- void openFrameReport ()
- void writeFrameReport (vector< FRAME > *listFrame0, vector< FRAME > *listFrame1)
- void closeReport ()

Public Attributes

- FILE * pReport

4.12.1 Detailed Description

Definition at line 9 of file REPORT.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 REPORT::REPORT ( ) [inline]

Definition at line 15 of file REPORT.h.

4.12.3 Member Function Documentation

4.12.3.1 void REPORT::closeReport ( )

Definition at line 19 of file REPORT.cpp.

4.12.3.2 void REPORT::openFrameReport ( )

Definition at line 4 of file REPORT.cpp.

4.12.3.3 void REPORT::writeFrameReport (vector< FRAME > * *listFrame0*, vector< FRAME > * *listFrame1*)

Definition at line 12 of file REPORT.cpp.

4.12.4 Member Data Documentation

4.12.4.1 FILE* REPORT::pReport

Definition at line 17 of file REPORT.h.

The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/REPORT.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/REPORT.cpp

4.13 TS Class Reference

`#include <TS.h>`

Public Member Functions

- TS (int PID, int flagAF, int sizeAF, int transErrorInd, int payloadUnitInd, int transPrior, int tranSacramb, int contCount, int flagNALU, list< PES > listPES, list< NALU > listNAL)
- int getPID ()
- int getPayloadUnitInd ()
- int getFlagAF ()
- list< PES > getListPES ()
- list< NALU > getListNALU ()
- int lossDetectionTS (int countAux)

Static Public Member Functions

- static TS readTS (char *pbuff)

4.13.1 Detailed Description

Definition at line 10 of file TS.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 TS::TS (int *PID*, int *flagAF*, int *sizeAF*, int *transErrorInd*, int *payloadUnitInd*, int *transPrior*, int *tranSacramb*, int *contCount*, int *flagNALU*, list< PES > *listPES*, list< NALU > *listNAL*) [inline]

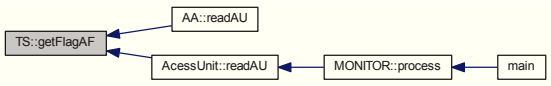
Definition at line 28 of file TS.h.

4.13.3 Member Function Documentation

4.13.3.1 int TS::getFlagAF ( ) [inline]

Definition at line 51 of file TS.h.

Here is the caller graph for this function:



4.13.3.2 list<NALU> TS::getListNALU ( ) [inline]

Definition at line 59 of file TS.h.

Here is the caller graph for this function:

4.13.3.3 list<PES> TS::getListPES ( ) [inline]

Definition at line 55 of file TS.h.

Here is the caller graph for this function:

4.13.3.4 int TS::getPayloadUnitInd ( ) [inline]

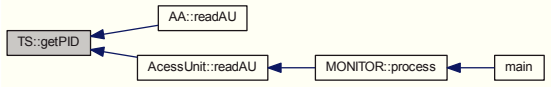
Definition at line 47 of file TS.h.

Here is the caller graph for this function:

4.13.3.5 int TS::getPID ( ) [inline]

Definition at line 43 of file TS.h.

Here is the caller graph for this function:



4.13.3.6 int TS::lossDetectionTS (int *countAux*)

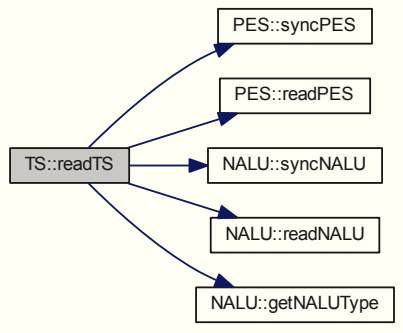
Definition at line 70 of file TS.cpp.

Here is the caller graph for this function:

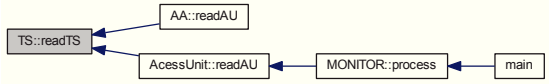
4.13.3.7 TS TS::readTS (char * *pbuff*) [static]

Definition at line 20 of file TS.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/TS.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/TS.cpp

4.14 VQA Class Reference

`#include <VQA.h>`

Public Member Functions

- VQA ()
- vector< FRAME > getListFrame0 ()
- vector< FRAME > getListFrame1 ()
- float clacVQ2D ()
- float clacVQauxView ()
- float clacVQ3D ()

4.14.1 Detailed Description

Definition at line 10 of file VQA.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 VQA::VQA ( ) [inline]

Definition at line 32 of file VQA.h.

4.14.3 Member Function Documentation

4.14.3.1 float VQA::clacVQ2D ( )

Definition at line 20 of file VQA.cpp.

4.14.3.2 float VQA::clacVQ3D ( )

Definition at line 36 of file VQA.cpp.

4.14.3.3 float VQA::clacVQauxView ( )

Definition at line 28 of file VQA.cpp.

4.14.3.4 vector<FRAME> VQA::getListFrame0 ( ) [inline]

Definition at line 34 of file VQA.h.

4.14.3.5 vector<FRAME> VQA::getListFrame1 ( ) [inline]

Definition at line 37 of file VQA.h.

The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/VQA.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/VQA.cpp

4.15 yyvrfv Class Reference

`#include <yyvrfv.h>`

Public Member Functions

- yyvrfv (void)
- ~yyvrfv (void)

4.15.1 Detailed Description

Definition at line 2 of file yyvrfv.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 yyvrfv::yyvrfv (void)

Definition at line 4 of file yyvrfv.cpp.

4.15.2.2 yyvrfv::~yyvrfv (void)

Definition at line 9 of file yyvrfv.cpp.

The documentation for this class was generated from the following files:

- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/yyvrfv.h
- C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/yyvrfv.cpp

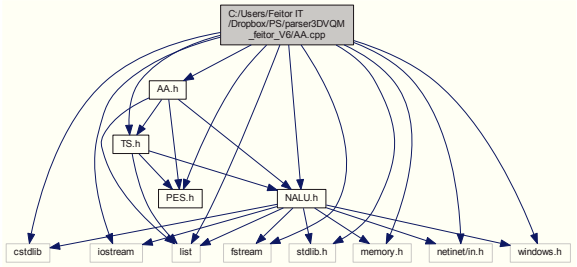
Chapter 5

File Documentation

5.1 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/AA.cpp File Reference

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <memory.h>
#include <netinet/in.h>
#include <windows.h>
#include <list>
#include "NALU.h"
#include "PES.h"
#include "TS.h"
#include "AA.h"
```

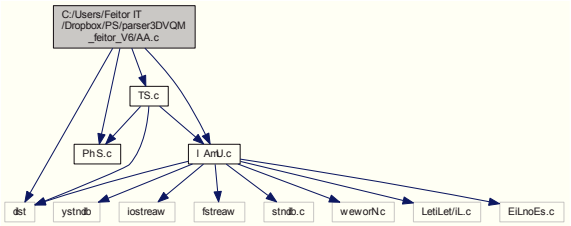
Include dependency graph for AA.cpp:



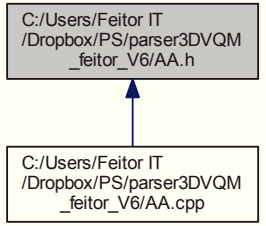
5.2 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/AA.h File Reference

```
#include <list>
#include "NALU.h"
#include "PES.h"
#include "TS.h"
```


Include dependency graph for AA.h:



This graph shows which files directly or indirectly include this file:



Classes

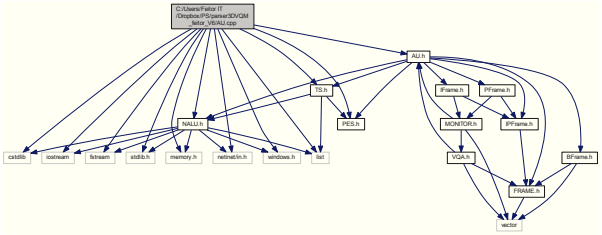
- class AA

5.3 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/AU.cpp File Reference

```
#include <cstdlib>
```

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <memory.h>
#include <netinet/in.h>
#include <windows.h>
#include <list>
#include "NALU.h"
#include "PES.h"
#include "TS.h"
#include "AU.h"
```

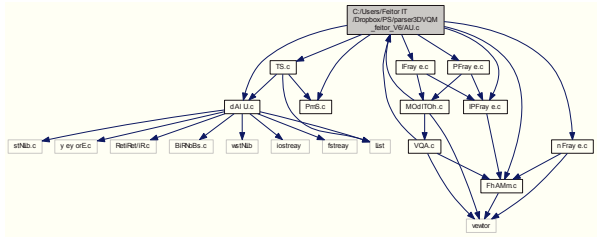
Include dependency graph for AU.cpp:



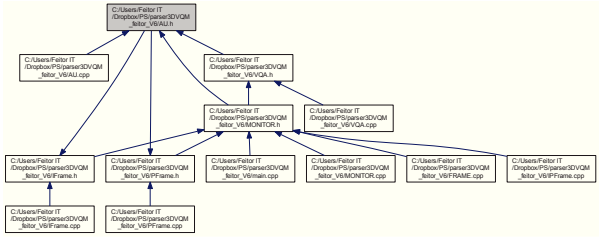
5.4 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/AU.h File Reference

```
#include "NALU.h"
#include "PES.h"
#include "TS.h"
#include "FRAME.h"
#include "IPFrame.h"
#include "IFrame.h"
#include "PFrame.h"
#include "BFrame.h"
```

Include dependency graph for AU.h:



This graph shows which files directly or indirectly include this file:



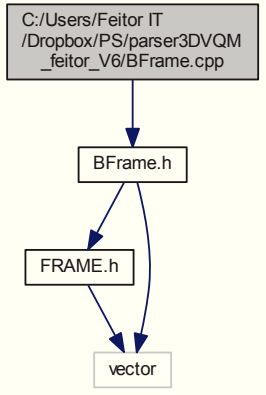
Classes

- class AcessUnit

5.5 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/BFrame.cpp File Reference

```
#include "BFrame.h"
```

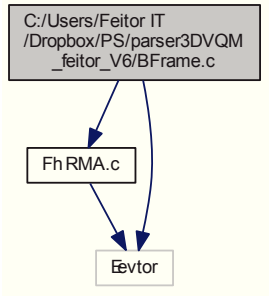
Include dependency graph for BFrame.cpp:



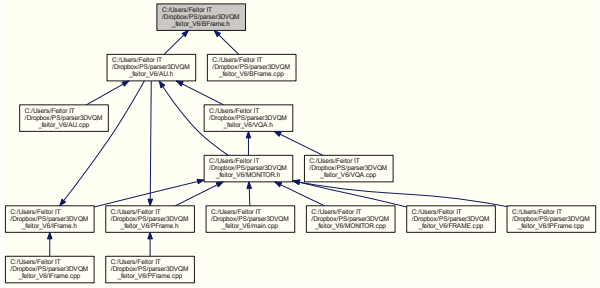
5.6 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/BFrame.h File Reference

```
#include "FRAME.h"
#include <vector>
```

Include dependency graph for BFrame.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BFrame

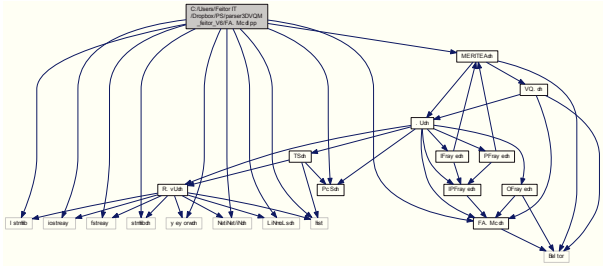
5.7 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/FRAME.cpp File Reference

```
#include <cstdlib>
```



```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <memory.h>
#include <netinet/in.h>
#include <windows.h>
#include <list>
#include "PES.h"
#include "FRAME.h"
#include "MONITOR.h"
```

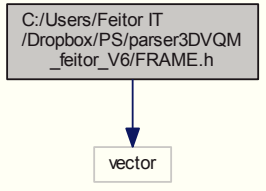
Include dependency graph for FRAME.cpp:



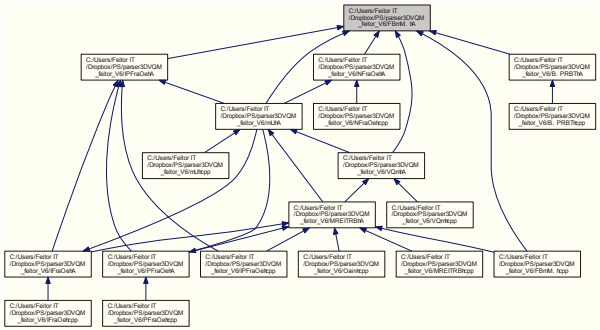
5.8 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/FRAME.h File Reference

```
#include <vector>
```

Include dependency graph for FRAME.h:



This graph shows which files directly or indirectly include this file:



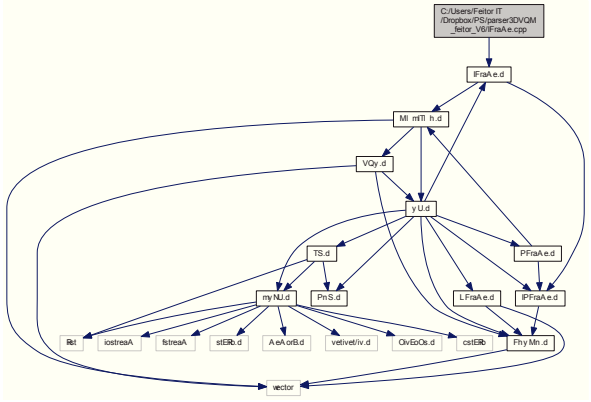
Classes

- class FRAME

5.9 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/IFrame.cpp File Reference

```
#include "IFrame.h"
```

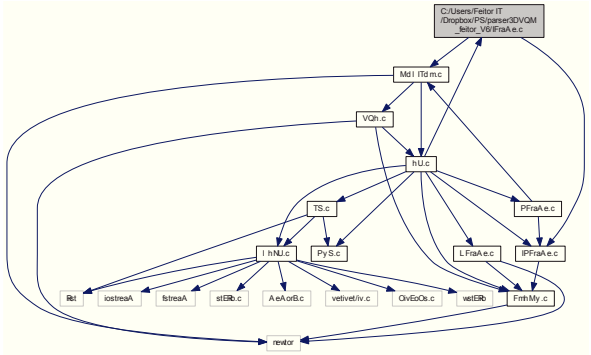
Include dependency graph for IFrame.cpp:



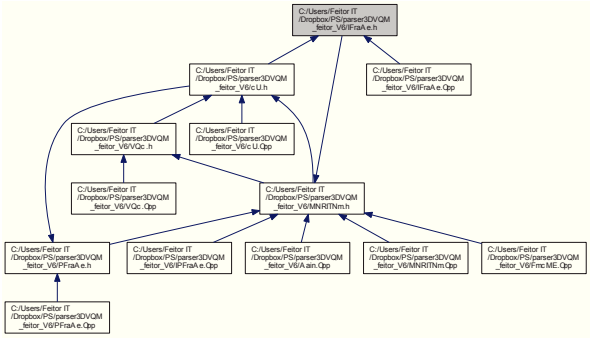
5.10 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/IFrame.h File Reference

```
#include "IPFrame.h"
#include "MONITOR.h"
```

Include dependency graph for IFrame.h:



This graph shows which files directly or indirectly include this file:



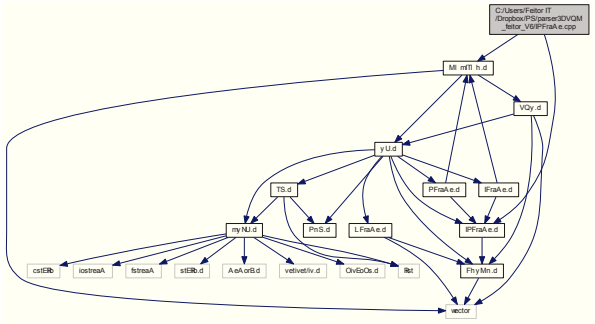
Classes

- class IFrame

5.11 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/IPFrame.cpp File Reference

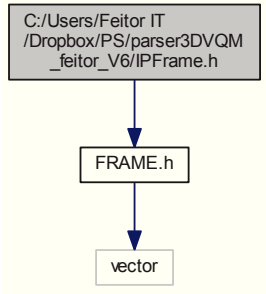
```
#include "IPFrame.h"
```

```
#include "MONITOR.h"
```
Include dependency graph for IPFrame.cpp:

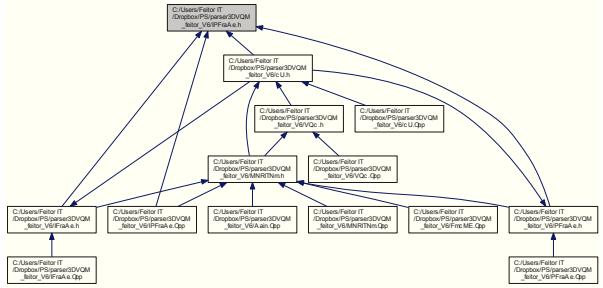


5.12 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/IPFrame.h File Reference

```
#include "FRAME.h"
```
Include dependency graph for IPFrame.h:



This graph shows which files directly or indirectly include this file:

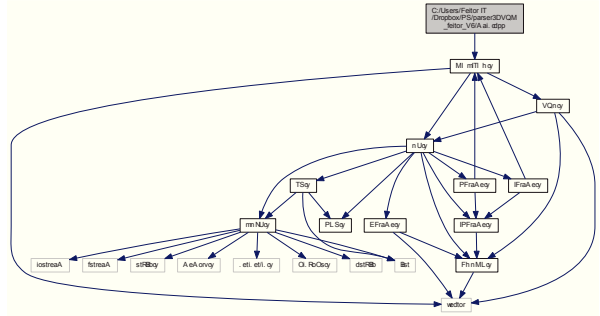


Classes

- class IPFrame

5.13 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/main.cpp File Reference

```
#include "MONITOR.h"
```
Include dependency graph for main.cpp:



Functions

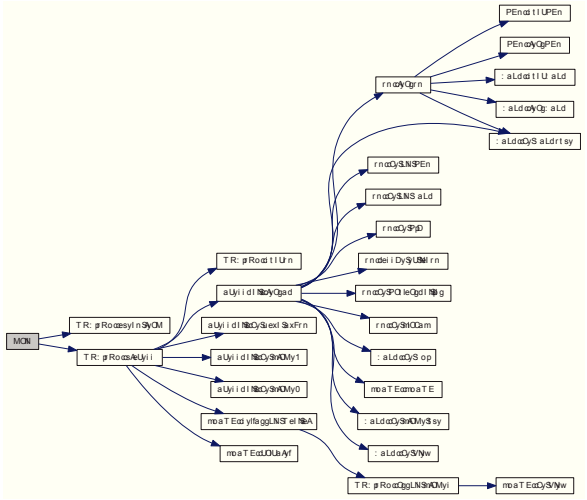
- int main (int argc, char **argv)

5.13.1 Function Documentation

5.13.1.1 int main (int *argc*, char ** *argv*)

Definition at line 6 of file main.cpp.

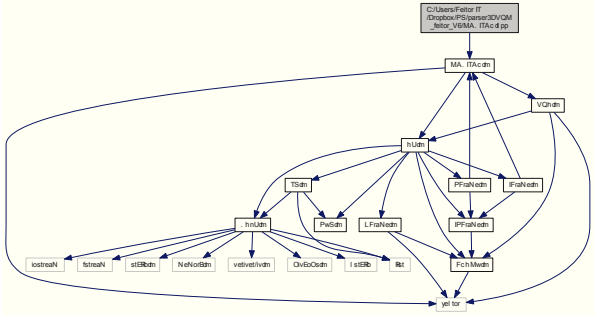
Here is the call graph for this function:



5.14 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/MONITOR.cpp File Reference

```
#include "MONITOR.h"
```

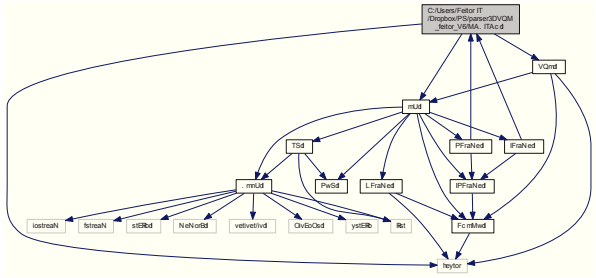
Include dependency graph for MONITOR.cpp:



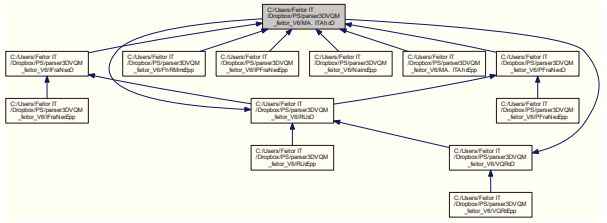
5.15 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/MONITOR.h File Reference

```
#include "AU.h"
#include "VQA.h"
#include <vector>
```

Include dependency graph for MONITOR.h:



This graph shows which files directly or indirectly include this file:



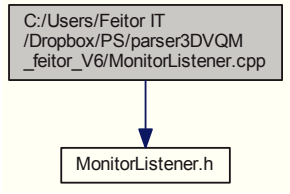
Classes

- class MONITOR

5.16 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/MonitorListener.cpp File Reference

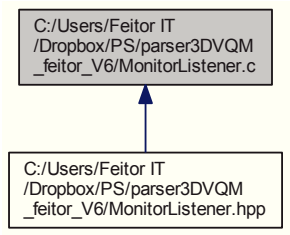
```
#include "MonitorListener.h"
```

Include dependency graph for MonitorListener.cpp:



5.17 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/MonitorListener.h File Reference

This graph shows which files directly or indirectly include this file:



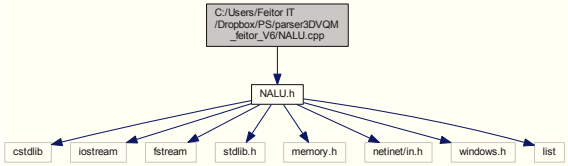
Classes

- class MonitorListener

5.18 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/NALU.cpp File Reference

```
#include "NALU.h"
```

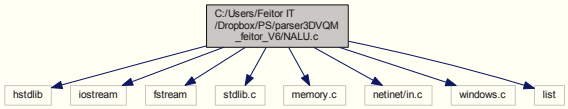
Include dependency graph for NALU.cpp:



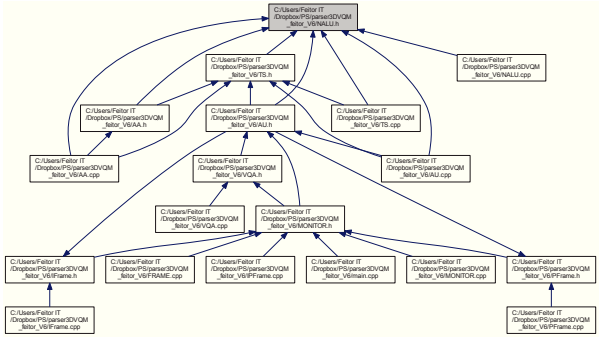
5.19 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/NALU.h File Reference

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <memory.h>
#include <netinet/in.h>
#include <windows.h>
#include <list>
```

Include dependency graph for NALU.h:



This graph shows which files directly or indirectly include this file:



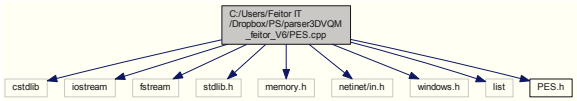
Classes

- class NALU

5.20 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/PES.cpp File Reference

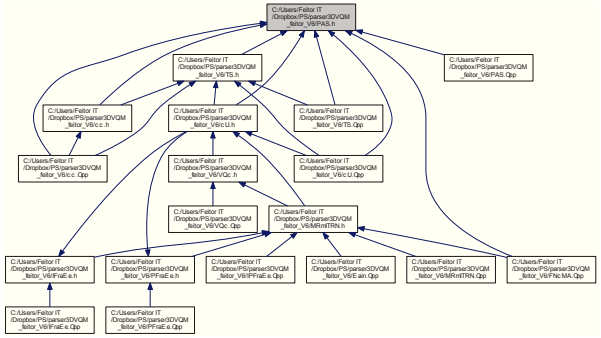
```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <memory.h>
#include <netinet/in.h>
#include <windows.h>
#include <list>
#include "PES.h"
```

Include dependency graph for PES.cpp:



5.21 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/PES.h File Reference

This graph shows which files directly or indirectly include this file:



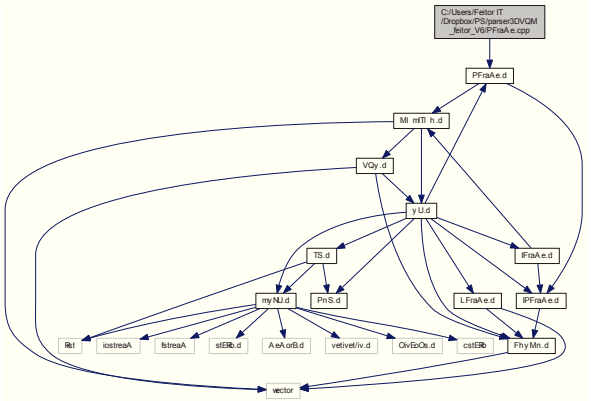
Classes

- class PES

5.22 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/PFrame.cpp File Reference

```
#include "PFrame.h"
```

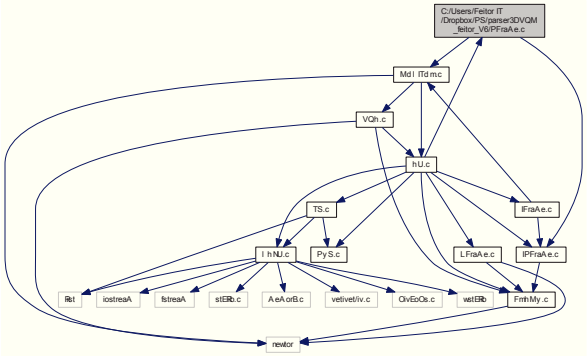
Include dependency graph for PFrame.cpp:



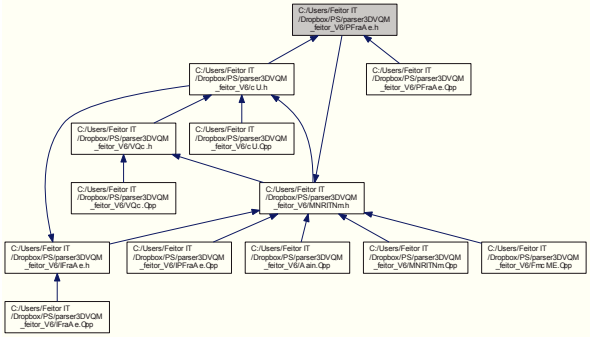
5.23 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/PFrame.h File Reference

```
#include "IPFrame.h"
#include "MONITOR.h"
```

Include dependency graph for PFrame.h:



This graph shows which files directly or indirectly include this file:



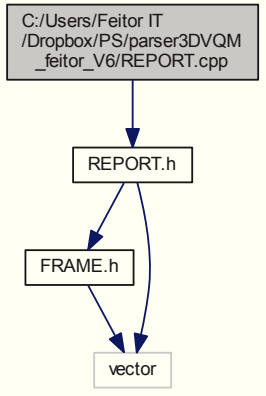
Classes

- class PFrame

5.24 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/REPORT.cpp File Reference

```
#include "REPORT.h"
```

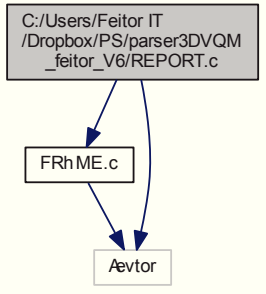

Include dependency graph for REPORT.cpp:



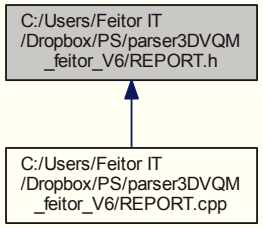
5.25 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/REPORT.h File Reference

```
#include "FRAME.h"
#include <vector>
```

Include dependency graph for REPORT.h:



This graph shows which files directly or indirectly include this file:



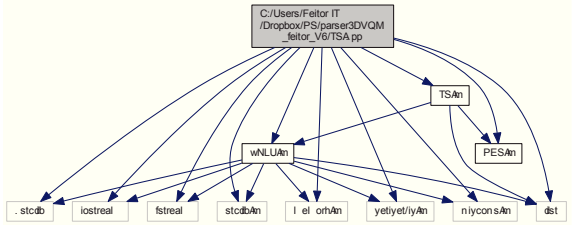
Classes

- class REPORT

5.26 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/TS.cpp File Reference

```
#include <cstdlib>
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <memory.h>
#include <netinet/in.h>
#include <windows.h>
#include <list>
#include "NALU.h"
#include "PES.h"
#include "TS.h"
```

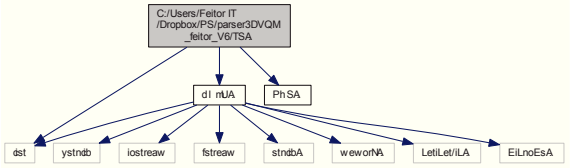
Include dependency graph for TS.cpp:



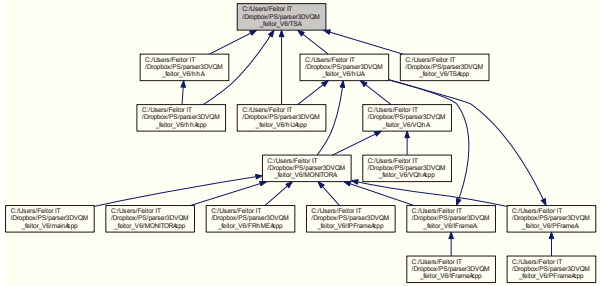
5.27 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/TS.h File Reference

```
#include <list>
#include "NALU.h"
#include "PES.h"
```

Include dependency graph for TS.h:



This graph shows which files directly or indirectly include this file:



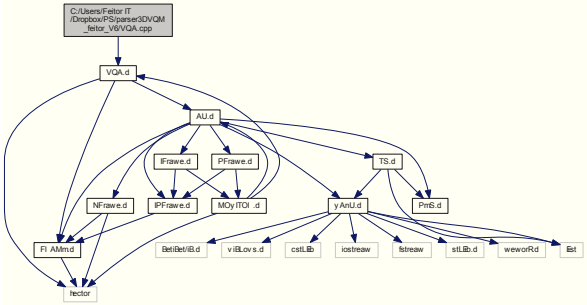
Classes

- class TS

5.28 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/VQA.cpp File Reference

```
#include "VQA.h"
```

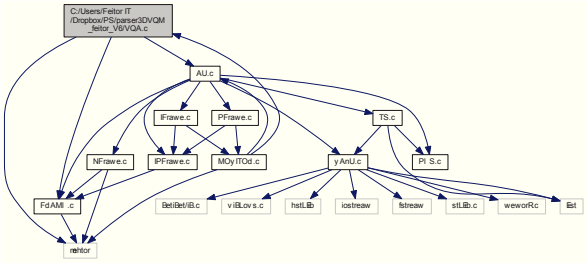
Include dependency graph for VQA.cpp:



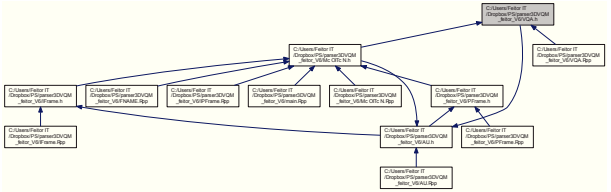
5.29 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/VQA.h File Reference

```
#include "AU.h"
#include "FRAME.h"
#include <vector>
```

Include dependency graph for VQA.h:



This graph shows which files directly or indirectly include this file:



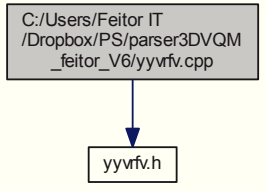
Classes

- class VQA

5.30 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/yyvrfv.cpp File Reference

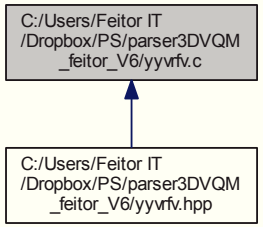
```
#include "yyvrfv.h"
```

Include dependency graph for yyvrfv.cpp:



5.31 C:/Users/Feitor IT/Dropbox/PS/parser3DVQM_feitor_V6/yyvrfv.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class yyvrfv