NTNU – Trondheim
Norwegian University of
Science and Technology

Video metric measurements in an FPGA for use in objective no-reference video quality analysis

Eirik Tørud Nordeng

Master of Science in Electronics
Submission date: June 2013
Supervisor: Kjetil Svarstad, IET
Co-supervisor: Jørgen Linnerud, Cisco

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

The purpose of this assignment is to examine the possibility of using an FPGA for extracting different metrics from a video stream. These metrics should provide information regarding the state of the hardware, meaning that faulty hardware and erroneous components should affect the metrics. Different algorithms are to be evaluated and the most promising algorithms should be implemented using VHDL code.

Abstract

This thesis presents a way of performing objective video quality analyses in order to point out faults in the hardware of a video system that uses analogue video transmission technologies. The approach focuses on performing simple digital processing and analyses of the video data coherently using an FPGA. Several metrics that correlates with specific distortions are developed. These metrics give good indications of the state of the video system components. The algorithms are tested using MATLAB and mapped to an FPGA. The key components are implemented and verified in VHDL, and synthesized for an Altera Cyclone II FPGA. The thesis concludes that the proposed system has the ability to discover board-level faults in a video system that utilizes an FPGA and analogue video transmission. The system also has the ability to supplement external quality assessment systems in most cases, and function as a good alternative in cases where a quick and simple assessment of a video system is desired.

Sammendrag

Denne rapporten presenterer et system for å utføre objektiv kvalitetsanalyse av video. Dette systemet skal påpeke feil i maskinvaren i et videosystem som bruker analoge videooverføringsteknologier. Tilnærmingen fokuserer på å utføre enkel digital prosessering og enkle analyser av video parallelt ved hjelp av en FPGA. Flere algoritmer som regner ut verdier som korrelerer med bestemte distorsjoner har blitt utviklet. Disse verdiene gir gode indikasjoner på tilstanden til videosystemers komponenter. Algoritmene er testet ved hjelp av MATLAB og implementert for bruk i en FPGA. De viktigste komponentene er implementert og verifisert i VHDL og syntetisert mot en Altera Cyclone II FPGA. Det konkluderes med at det foreslåtte systemet har evnen til å oppdage feil i maskinvaren i et videosystem som anvender en FPGA og analog videooverføring. Systemet har også muligheten til supplere eksterne systemer for kvalitetsanalyse i de fleste tilfeller, og det vil fungere som et godt alternativ til disse i de tilfeller der en rask og enkel vurdering av et videosystem er ønskelig.

Preface

This thesis is written at the Institute of Electronics and Telecommunications (IET) at the Norwegian University of Science and Technology (NTNU) during the spring of 2013. The assignment is given by Cisco Norway. The work has been done under the guidance of Professor Kjetil Svarstad at NTNU and Jørgen Linnerud at Cisco Norway. The thesis is a continuation of my specialization project which served as a preparatory study. I chose this assignment because of its ties to image processing and FPGA development. I wish to thank Kjetil and Jørgen for their many guidance hours and plenty of good advices.

Eirik Tørud Nordeng
Trondheim, June 2013

Contents

List of Figures

List of Tables

Abbreviations

ADC	**A**nalog to **D**igital **C**onverter
ASIC	**A**pplication **S**pecific **I**ntegrated **C**ircuit
ASSP	**A**pplication **S**pecific **S**tandard **P**roduct
DAC	**D**igital to **A**nalog **C**onverter
DAC	**D**iscrete **C**osine **T**ransform
DFF	**D**ata/**D**elay **F**lip-**F**lop
DSP	**D**igital **S**ignal **P**rocessor
EOF	**E**nd **O**f **F**rame
EOL	**E**nd **O**f **L**ine
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray
FRVQA	**F**ull-**R**eference **VQA**
HDL	**H**ardware **D**escriptional **L**anguage
HVS	**H**uman **V**isual **S**ystem
IC	**I**ntegrated **C**ircuit
IP	**I**ntellectual **P**roperty
LE	**L**ogical **E**lement
LSB	**L**east **S**ignificant **B**it
LUT	**L**ook**U**p **T**able
MSB	**M**ost **S**ignificant **B**it
MSE	**M**ean **S**quare **E**rror
NRVQA	**N**o-**R**eference **VQA**
PLL	**P**hase **L**ocked **L**oop
PSNR	**P**eak **S**ignal to **N**oise **R**atio
PWM	**P**ulse-**W**idth **M**odulation
RAM	**R**andom-**A**ccess **M**emory

RRVQA	**R**educed-**R**eference **VQA**
SAD	**S**um of **A**bsolute **D**ifferences
SSD	**S**um of **S**quared **D**ifferences
VQA	**V**ideo **Q**uality **A**ssessment
VQM	**V**ideo **Q**uality **M**etric

Chapter 1

Introduction

This report and the work related to it is the product of the master thesis of Eirik Tørud Nordeng at the Circuit and Systems group of the Department of Electronics and Telecommunications (IET) at the Norwegian University of Science and Technology (NTNU). The work is a continuation of the specialization project [1] which served as a preparatory study. The following sections present the assignment specifications and the structure of the report.

1.1 Assignment specification

Objective video quality analysis means measuring the quality of a video stream by utilizing a system running one or more VQA (Video Quality Assessment) algorithms. The algorithms produce outputs that correlates with the level of distortion in a video stream. The term no-reference quality analysis means that there is no original version of the video signal available for comparison to the VQA system. When designing such metric systems, one therefore has to assume that the video stream has some certain properties, or simplify the analysis by feeding the system with known test patterns. This work focuses on pure no-reference analyses, but investigates the use of such test patterns and their effects on the metrics as well.

The objective of the assignment is to develop video quality metrics for detecting artifacts that are results of board-level failures. It is therefore natural to look for artifacts related to analogue transmissions and physical defects such as stuck-at-faults and faulty filter components. Although it is not the main objective, it is also of interest to detect errors caused by faults in digital systems, and faults that are introduced in the design phase.

The intended application for the system is testing at the production site, where it can be beneficial to not have to rely on expensive external equipment or a person with a trained eye. In-house regression testing is also a possible application for the system, as it can help speed up the design process. It is important to notice that this system is not intended as an alternative to complex external quality assessment systems, but rather as a quick and simple supplement. The desire is to be able to quickly configure an on-board FPGA for analysing a video stream and point out known distortions and artifacts related to board-level faults, as figure 1.1 shows. The system should input a 24-bit RGB video signal with explicit synchronization signals and output the video quality metrics as a range of values consistent with the developed metric models.

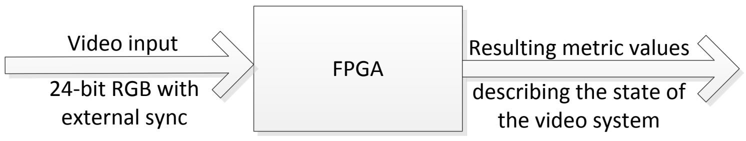

Video input
24-bit RGB with external sync
FPGA
Resulting metric values
describing the state of the video system

Figure 1.1: Simple overview of the desired video quality assessment system.

The assignment is divided into two parts. The first part, which was carried out during the specialization project, involved studying different methods for performing qualitative analyses of run-time video. This included a clear overview showing the use of resources, the estimated real time capabilities of the systems and an analysis of the benefits and drawbacks regarding mapping of the different algorithms to an FPGA. The second part, which is the focus of this thesis, involves modeling and writing HDL modules for measuring the video metrics based on the procedures and approaches found in the first phase of the project. All VQA models are to be tested thoroughly using different images with specific properties. In the end, the project should result in several suggestions for VHDL modules that can be used for evaluating a video stream in real time using an FPGA with and without knowledge of the input signal.

1.2 Contribution

Earlier work related to objective no-reference analogue video metric measurements is concerned about visual distortions in video. These metric systems are designed solely to provide information related to how well a human can perceive distortions, and does not go into the underlying cause of the faults. The contribution from this assignment is a set of video quality analysis systems that are designed to uncover physical failures on the board-level of a video system. In addition, the video metric measurements are developed specifically for use coherently in an FPGA. This provides the possibility for using several simple metrics together in order to discover failures in a video system. The algorithms and their VHDL models are tested and analysed thoroughly in order to ensure that they work as intended.

1.3 Report layout

The approach taken for solving the task at hand is presented in chapter 2, together with a review of relevant previous work.

Chapter 3 presents the theory that is necessary for understanding how video, image processing and video quality metrics work. It also presents basic information on relevant electronic systems and their relations to video artifacts. Chapter 4 presents the tools used for developing the algorithms and their implementations. The target electronics are also presented here.

The developed VQA systems are presented in chapter 5, and models of these systems are presented in chapter 6, together with the approach for testing them. The implementation and verification of the FPGA-specific algorithms is presented in chapter 7.

The assignment results can be found in chapter 8. This includes results from testing the models and results from synthesizing the circuits described in VHDL. An analysis of these results and an analysis of the metrics in general can be found in chapter 9, while some concluding remarks are found in chapter 10.

All results from testing and modeling in MATLAB can be found in appendix B, and the scripts modeling the systems are included in appendix A.

Chapter 2

The approach and previous work

This chapter presents the approach taken when working with this project, together with the results and contributions from relevant previous work.

A literature study concerning no-reference video quality analysis was presented in the authors specialization project, which preceded the work on this thesis. In the specialization project, the methods for calculating many previously developed no-reference metrics were analysed and the possibilities for performing the calculations on an FPGA were evaluated. The results from this study has formed the background for the video quality metrics presented later in this report.

The evaluated metrics were based on several different approaches, ranging from methods based on spatial filtering to complex statistical models and analyses in other domains. All of the metrics were developed for running in software, which meant that an analysis of the possibilities for mapping the algorithms to an FPGA was essential. In addition, many of the newly developed video metrics were designed for discovering artifacts related to digital video and the HVS (Human Visual System). An important part of the work was therefore to distinguish the essential algorithm components from the processing related to these issues.

The results from the analyses point towards creating many simple algorithms that together say a lot about the state of the board components. Since the algorithms are to run on an FPGA, the possibility for resource sharing, i.e. using the same modules for calculating different metrics, is important. Expensive transformations and complex computations are undesirable since such components may demand a high amount of resources.

DSPs (Digital Signal Processors) are normally the desired platform for metric measurements of video quality. However, since FPGAs are usually used for preprocessing of video, they are often the first component in the video processing system. They are also inherently suited for parallel processing of image data, providing the possibility for calculating several video metrics in real time without influence from other processing devices.

The combination of real time processing on FPGAs, and focus on discovering distortions that can be traced back to physical board-level faults using simple metric calculations, provide a new perspective on the field of objective no-reference video quality analysis. Although this work does not present a fully developed video quality analysis system, the model simulation and synthesis results indicates that the proposed system will function as a valuable addition to production testing and regression testing in the production and design phase of new video systems.

The work with the project is divided into three sections. First, the algorithms are developed based on video and image processing, knowledge of the target platform and the results from the specialization project. The video input used for testing is based on the VGA standard. Second, the algorithms are modelled in MATLAB and tested using a series of specially developed tests. The last part of the work consists of implementing the key algorithm components for evaluating their resource consumption and their real time capabilities.

The conclusion is presented based on the results from this approach. It gives a clear indication on whether such a system is usable and desirable compared to already existing systems on other platforms.

Chapter 3

Theoretical background

The following sections will take a look at the theory behind video quality assessment systems. This includes theory of electronic video systems, video transmission and digital signal processing. Some relevant previously developed algorithms are also presented.

The chapter begins by presenting properties of the field-programmable gate array and the advantages and limitations of this device compared to other signal processing platforms. Next, the representation of video in a computer is examined. Since most VQA systems use digital signal processing in order to output a good metric, some essential processing principles are described, before some previously developed VQA algorithms are presented. Because it is important to know how to relate distortions to specific artifacts, the last section examines how physical faults can lead to specific artifacts in analogue video transmission.

3.1 Field-programmable gate arrays

An FPGA (Field-Programmable Gate Array) is an electronic device that can represent functionality as a digital circuit. FPGAs are especially well suited for low level image processing since they are inherently parallel and therefore easily can exploit the parallelism that is present in images [2, Chapter 2].

The smallest unit in an FPGA is the logic cell [2, Chapter 2]. This is a basic building block that normally contains a LUT (LookUp Table) and a DFF (Data Flip-Flop). By combining several logic cells through router blocks, digital circuits are formed. The logic cells in Altera's devices are called LEs (Logical Elements) [3].

FPGAs are heavily bound by the memory bandwidth and therefore rely greatly on internal memory known as block RAM. In order to properly exploit the capabilities of an FPGA, most systems therefore rely only on a small amount of data, of which there is room for in the target FPGA. This is especially significant in image processing, since most FPGAs will have problems buffering an entire image frame, and instead have to rely on smart memory techniques and processing in order to stay within the resource specifications [2, Chapter 5].

Timing constraints is another important issue, especially in real time video and image processing. It is required that all of the computations in a system have been finished before the next clock cycle. However, if the computations are to intense and cannot be completed within the required period, one can resort to pipelining, which is a technique where logic is split into smaller blocks, spread over multiple clock cycles. Pipelining can be utilized at most levels in the circuit, but may introduce the need for some extra control logic, as the timing of the data is being altered. As with all forms of data buffering, pipelining increases the latency of the system [2, Chapter 5].

Sometimes there is a necessity of different clock domains in an FPGA system. Multiple clock domains are used in video and image processing when it is required that the pixel data is processed at a different rate than the rate of the arriving pixels. Employing such techniques allows for more elaborate processing and better interfacing with external devices. However, it does make designing the systems harder, since communication and passing of data between the different domains requires complex synchronisation components [2, Chapter 5].

For a more thorough examination of FPGAs, see the report from the preliminary study [1, Chapter 1] or Compton's survey on FPGAs [4].

3.2 Video representation

In computers, a video signal is represented as a stream of images. The images are represented as a number of lines consisting of a number of picture elements, or pixels. The pixels are formatted in a specific way, for instance in the additive RGB colour space or in the YCbCr colour space. By changing the values of the pixels, all available colours are created [5, Chapter 3].

VGA (Video Graphics Array) has for a long time been a very popular standard for transmitting video between computers and from computers to monitors. It is currently used as a high-resolution video standard in some computer and consumer equipment, where the ability to transmit a sharp, detailed image is essential [5, Chapter 2].

The VGA standard transmits the video as three separate colour components representing red, green and blue. The components are transmitted using separate wires together with horizontal and vertical synchronization signals called *end of line* (EOL) and *end of frame* (EOF). An active resolution equal to 1920×1080 of a 1080p digital RGB signal, as defined within BT.709 and SMPTE 274M[1], has a sample rate of 148.5 MHz and a frame rate of 60 Hz [5, Chapter 4].

A VGA system divides the video frames into single rows. These rows are transmitted in sequence, using analogue representations of the pixel values. The sampling of the component values using an ADC are triggered using the pixel clock. When the end of a row is reached, EOL goes high, and after a short period, this signal is set low again and the next row of pixel values is transmitted. When an entire frame worth of rows has been transferred, EOF goes high. When EOF is set low, the next frame is to be transmitted in the same way as the previous one.

For a more comprehensive description of the VGA standard and other ways of representing video, see the report from the preliminary studies [1].

[1]The SMPTE (Society of Motion Picture and Television Engineers), together with the EBU (European Broadcasting Union), have presented the main parameters for coding, filtering and timing of video [5, Chapter 4]. The BT.709 and the SMPTE 274M present the parameter values for the HDTV standards and defines the 1080 line high definition video formats, respectively.

3.3 Digital signal processing

When the analogue video representation has been sampled, it is possible to employ digital processing on the resulting digital signal. Since a video stream is inherently a stream of images, it has been considered desirable to use image processing on single frames to obtain the required information. The image processing components that are significant for the work presented in the following chapters are described below.

3.3.1 Convolution

A popular way of looking at the spatial correlation of images is by employing a convolution filter. Convolution is a form of linear spatial filtering which is performed by moving a filter mask (a template) over the image and computing the sum of products at each location [6, Chapter 3.4]. A 2-dimensional convolution, or spatial convolution, is shown in equation 3.1, and its mechanism is shown in figure 3.1.

$$m(x,y) * i(x,y) = \sum_{k=-w}^{w} \sum_{l=-w}^{w} m(k,l)i(x-k,y-l) \tag{3.1}$$

The filter mask in figure 3.1 is a matrix consisting of nine cells. The template values of the filter are set static according to a known working pattern, resulting in a specific filter behaviour. The input of the filter is the nine pixels from the image. As shown in the figure, these elements are spatially correlated in that they are neighbours. The filter mask moves through the image from left to right, one row at a time until the whole frame has been filtered. It then starts over again with the next frame.

A problem occurs when the border values of the image are to be filtered. The filter mask will then exceed the image borders, which causes unknown behaviour. In order to reduce these issues, the border of the image can be padded with values according to for example the border pixel values or a constant value [6].

The implementation of this component for an FPGA requires buffering of two image rows, in addition to a circuit performing the filter computations. One of

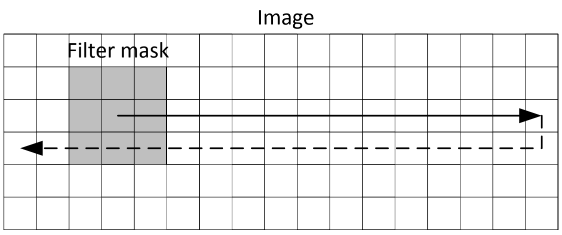


FIGURE 3.1: The mechanics of linear spatial filtering using a 3×3 filter mask.

the more efficient ways of doing this is by using a row buffer component connected to a filter component [2, Chapter 8]. A parallel row buffer consists of two shift registers for the buffering of the rows and three shift registers for the 3×3 filter mask.

3.3.2 Median filter

Another popular filter type is the median filter. This is an order-statistic nonlinear filter which is heavily used for reducing the amount of salt-and-pepper noise in images (see section 3.5.3). A median filter finds the median in a set of values by sorting them and outputting the value that is greater than or equal to half the values and smaller than or equal to the other half [6, Chapter 3].

The filter mask moves through the image in the same way as the convolution filter, but the filter computation is significantly different. The nine values must be sorted in order to find the median. This should not be done iteratively since the system is to run on an FPGA. Instead, an architecture relying on parallel processing can be utilized. A common way of doing this is by using combinatorial sorter components [7]. A two-input sorter component is made from a magnitude detector, which detects whether the one input is larger than the other, and a swapper, which swaps the two inputs according to the output from the magnitude detector. As shown in figure 3.2, by combining three pipelines and three instances of this component, a three-input sorter can be created.

A functional median filter is achieved by combining these sorters with the row buffer and two pipeline stages, as shown in figure 7.6 [7]. The result is a cleverly

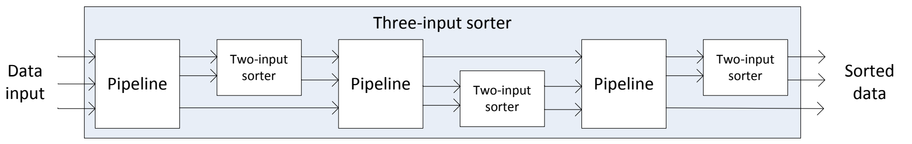


FIGURE 3.2: The architecture of a three-input sorter used in median filter components. The figure is adapted from [7].

pipelined module that filters the image with a latency of 11 clock cycles for the filter computation alone.

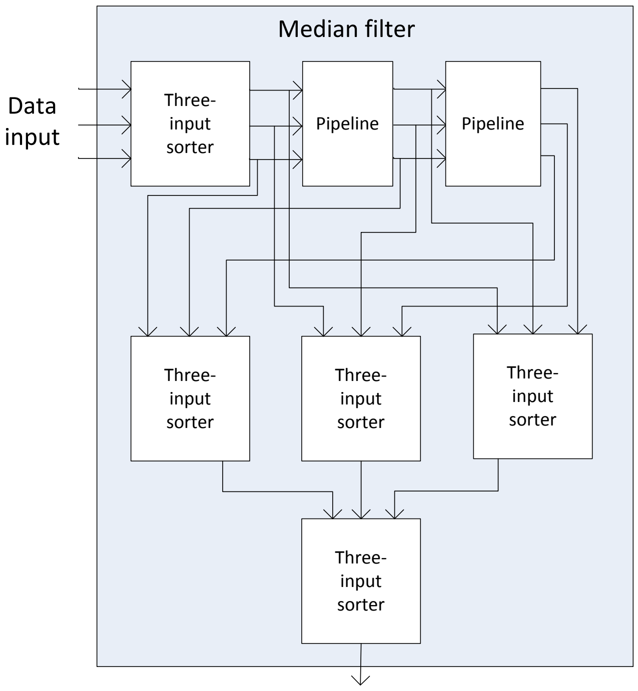


FIGURE 3.3: A common median filter architecture for utilization in an FPGA. The figure is adapted from [7].

3.3.3 Histogram processing

A histogram is developed by collecting data for representing the distribution of intensity levels in an image. Using histograms allows for visualising properties such as the brightness level and the level of contrast in an image [6, Chapter 3]. Figure 3.4 shows a grey scale representation of the red pixels in the Lena photography and its corresponding histogram.

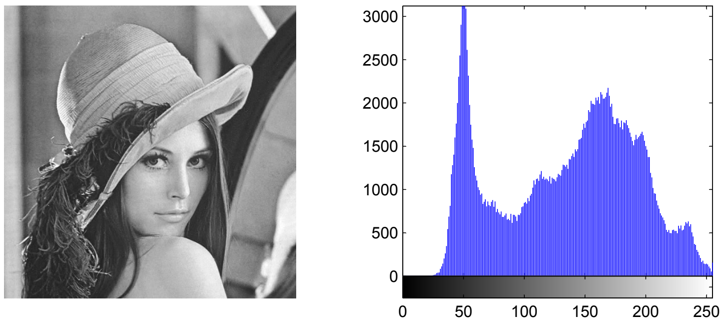


FIGURE 3.4: A histogram with 256 bins showing the values of the red picture elements in the Lena photography.

Histograms can present properties of an image that are hard to quantize otherwise. For example, a histogram where the values are contained at its lower parts signifies that the image is dark. In the same way, higher pixel values means that the image is bright [6, Chapter 3]. Figure 3.5 shows two examples of this.

A histogram can also help in determining if an image is high or low in contrast. If the image contrast is high, the values in the histogram will be spread out and be close to uniform, as shown in figure 3.6. In the opposite case, a low contrast image results in a histogram values that are contained in a small, spiked area [6, Chapter 3].

The architecture of a histogram accumulator consists of a decoder, an array of counters and a multiplexer for choosing which counter value to output [2, Chapter 7][2]. The image data serves as input to the decoder, which enables one of the counters. There is one counter per bin in the histogram, meaning that when data is input, the one counter belonging to the respective bin is enabled. The counter values can be read out in parallel or one by one using the multiplexer. The latter

[2]There is also an alternative way of realising a histogram accumulator. A design that utilizes dual-port memory needs fewer resources, but requires sequential resetting of the register. This demands a fast processing clock, which is not desirable in this system [2, Chapter 7].

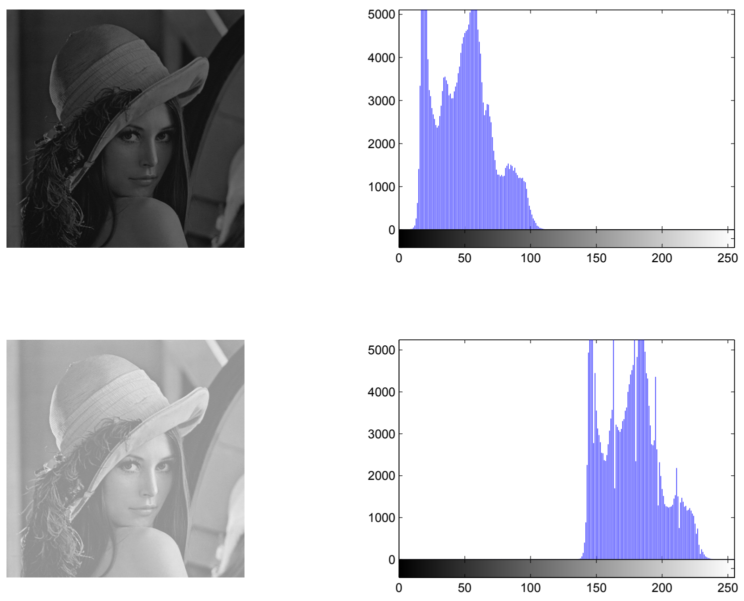


FIGURE 3.5: Histograms showing the values of the pixels in darkened and lightened grey scale versions of the red channel in the Lena photography.

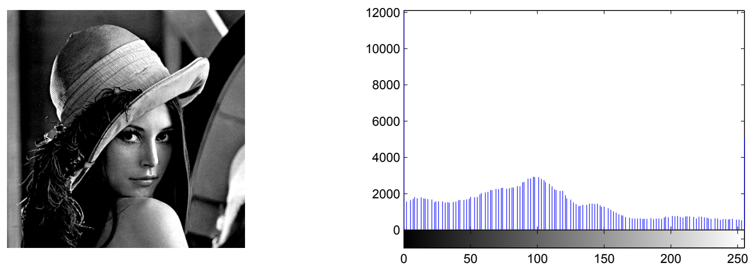


FIGURE 3.6: A histogram showing the values of the picture elements in the red channel of the Lena photography when the contrast is increased.

does however require an amount of clock cycles equal to the number of bins in the histogram.

3.4 Video quality assessment

A VQM (Video Quality Metric) can be based on both subjective and objective data [8, Chapter 3.4]. A subjective VQA system looks at how video is perceived by a number of test viewers. The output of such an assessment is a metric that is based on how the viewers perceived specific distortions or the overall quality of the video. An objective VQA system is based solely on an algorithm running on a computer. For the rest of this report, all mentioned VQA systems are objective.

3.4.1 Video quality metrics

After video started becoming popular in electronic devices, a lot of effort was put in to creating metrics for determining the quality of video. This has resulted in a large range of both simple and very complex metrics. Some of the more advanced metrics are based on the HVS (Human Visual System) and complex statistical analyses [8]. Although these metrics often are the most accurate ones, many simple quality metrics can achieve usable results by exploiting familiar knowledge about the general nature of some specific artifact. Artifacts such as blurring can for example be measured by analysing the sharpness of an image. The sharpness, which is the opposite of blurriness, can be measured by looking at spatial characteristics such as the image edge properties [8, Chapter 3].

The VQM that is most often utilized for determining the quality of an image is the PSNR (Peak Signal-to-Noise Ratio) [8, Chapter 3.4]. This metrics looks at the differences between the distorted image and the original version of the image. The metric is calculated by finding the ratio between the squared maximum data value and the MSE (Mean Square Error). It is usually expressed in the decibel scale because of the high dynamic range. Because PSNR is a widely used metric that is easily related to, it is often used as a reference for other metrics. The ways of calculating the metric is shown in equation 3.3, where I is the original image, $\tilde{I}$ is the distorted image and m is the maximum value of one pixel.

$$MSE = \frac{1}{XY}\sum_X\sum_Y\left[I(x,y) - \tilde{I}(x,y)\right]^2 \tag{3.2}$$

$$PSNR = 10\log\frac{m^2}{MSE} \tag{3.3}$$

Since PSNR relies on the original image, it is characterized as an objective Full-Reference VQA (FRVQA). Unfortunately, by operating solely on a pixels-by-pixel basis, this measurement neglects the important influence of the image content [8, Chapter 3]. The requirement of having access to the original image and the limitations caused by the simplicity of the measurement therefore makes it undesirable for use as a metric for measuring most artefacts and distortions, especially in real time systems.

VQA systems that are not reliant on a copy of the original image are called No-Reference VQAs (NRVQA). These systems are often harder to develop and have to rely on certain assumptions about an image. The assumptions are often based on parameters such as the dynamic range of the frames, the frequency bandwidth or statistical or spatial correlations [9–15].

A VQM from a NRVQA based on natural images is sometimes very hard to analyse because of the vast differences that can occur between the images. In a video sequence, changing between such images will cause the metric values to jump heavily and it will be impossible to see any correlation between them. Therefore, in many cases, the resulting metric provided by NRVQA systems can be analysed more easily if some demands can be made regarding the video input. This can for example be the requirement that the video input must consists of one static frame.

Many new NRVQA systems have to rely on using a lot of resources in order to find the quality of a video. Expensive transforms such as wavelets [12], DCTs [13] and colour space conversion, and complex statistical models [13, 16–18] are often utilized. Because most of these algorithms are designed for software, they are also inherently composed of highly iterative solutions with high memory footprints. Additionally, many of the metrics are designed for quantifying the perceptual quality of video and are therefore based on human perception[9, 10], which is rarely efficient when looking for all distortions caused by physical faults.

For a larger survey on some interesting NRVQA systems, see the report from the preliminary studies [1, Chapter 3].

When it is possible to feed a specific input into the VQA system, it is often desirable to increase the performance of the systems by actively relating the algorithms to this input. By using such signals, one can test for specific values in certain areas of a frame without having to buffer the entire original frame in the device. In this report, systems utilizing such methods are referred to as Reduced-Reference VQA (RRVQA) systems.

3.5 Electronic circuitry

There are many different concepts of video systems, including large variations of design techniques. However, all modern video systems rely on electronics for transferring visual information from one point to another. The transfer distance varies greatly, and consequently so does the technology used for different scenarios. VGA uses a technique where digital representations of video signals are converted to analogue signals using a DAC, before transmitting this signal through a medium such as a VGA cable. At the other end of this medium, the analogue signal is sampled by an ADC and fed into the core video system, where it is presented to a user or processed according to the nature of the system. There can also be some digital pre- and post-processing related to this transmission.

There are many cases where unwanted behaviour and deviations can be introduced into such a system. The failures come from unexpected behaviour internal to the system, which somehow manifest themselves in its external behaviour. The algorithmic or mechanical causes of these failures are called faults, while the problems themselves are called errors [19, Chapter 2]. The faults can be permanent, transient or intermittent, making them occasionally very hard to discover.

3.5.1 Fault sources

The typical faults that occur in a system utilizing VGA transmission revolves around the analogue representation of the video signal [20] (when distortions caused by digital compression issues are excluded). Since the data is transferred as analogue values, it is receptive to any kind of electrical interference caused by stray electromagnetic fields. It is also susceptible to any kind of errors caused by faults in physical components such as analogue filters, voltage sources and oscillators. Table 3.1 shows some of the problems one may encounter when designing and testing video systems.

Table 3.1: Overview of possible faults caused by physical defects and unwanted interference.

Fault	**Example**
Electrical interference	Sporadic, periodic and consistent random noise.
Faults in passive component values	Failure in analogue filters and slow signals.
Clock jitter	Faulty sampling of values.
Faulty voltage circuits	Sporadic failure and problems with reaching all voltage levels.

3.5.2 Associated video distortions

The faults mentioned in table 3.1 are believed to be able to manifest themselves into a range of distortions in a video stream. It is assumed that the types of faults can appear as different types of random noise, and as specific types of artifacts, as described below. Some of these assumptions are based on personal communication with engineers at Cisco [21].

If the components of a filter are faulty, for instance if the value of a component is erroneous or if a component is missing or broken, one might experience that the analogue signal is less resistant to noise, or that it is losing high frequency components. This leads to noise and blurring, respectively [20, 21].

If there are large deviations from the periodicity of a clock signal, hereinafter referred to as clock jitter, there will be problems with the sampling of the analogue signals. Erroneous sampling can create distortions that resemble noise, but has spatial correlations with the neighbouring samples. This effect can both blur the image and create a noisy type of distortion.

Problems with the voltage levels in the electronic circuitry can manifest themselves into sporadic failures of certain ICs or ASSPs, or there can be problems reaching certain voltage values [21]. The first problem can appear as sporadic erroneous values in the video stream, while the latter would result in the lack of certain pixel values, such as very small and very large values.

The analogue system is believed to be susceptible mostly to additive noise with a Gaussian probability distribution [22, Chapter 4.5]. However, such as speckle noise and salt and pepper noise are also believed to occur. If there is interference from electronic devices driven by for example a PWM (Pulse Width Modulator), the interference may appear as vertical or horizontal lines in the frame. These lines appear sporadic, according to the frequency of the device.

Jitter or failures linked to vertical and horizontal sync signals are common in many analogue video systems [21]. An erroneous vertical or horizontal synchronization signal results in displacement in the frame, by a scrolling[3] or a slanting effect[4], respectively [20].

Failures due to stuck-at-faults in the circuit or in ICs may also appear [21]. These faults can lead to failures such as difficulties in reaching specific values, and can be both sporadic, periodic and consistent due to the several possible states of a digital component.

Table 3.2 shows an overview of the faults and distortions discussed above. It is important to notice that most of the faults described here may or may not result in distortions that are easily noticed in the video. The nature of the faults and the complexity of the surrounding system determines if the failures surface as highly visible distortions, small and nearly undetectable artefacts, or if they surface at all.

In addition to failures caused by physical faults, failures related to digital processing of a video signal will cause distortions in a video stream. The most common type of digital distortions are blocking and blurring artifacts [8, 23]. The video coding standards that rely on motion compensation and block-based DCTs are especially susceptible for blocking artifacts. Blurring is caused by the loss of high

[3]The start of the frame appears further down on the screen, while the bottom section appears at the top.

[4]Imagine that the image consists of a text box. If there are faults in the horizontal sync signal, the text will appear as if it were in *italic*.

TABLE 3.2: Overview of possible failures that are caused by physical faults.

Problem	**Example of distortion**
Consistent random noise	Salt and pepper noise, multiplicative noise and Gaussian noise.
Periodic random noise	
Sporadic random noise	
Passive component faults	Blurring of the individual frames due to slowly changing values on the channels.
Clock jitter	Seemingly random noise.
Slow signals	Bad sampling of colour values. The colours are distorted.
Voltage problems	Bad sampling of colour values. The brightest and darkest colours can not be reached.

frequency information as a result of quantisation. For a list of common artifacts caused by digital video compression, see chapter 3.2 in "Digital Video Quality - Vision Models and Metrics" by Stefan Winkler [8]. This report will not cover any more artifacts caused by digital processing, although one must remember that artifacts caused by digital compression algorithms will affect most metrics.

3.5.3 Image noise models

Since the focus of this report is to discover artifacts related to analogue transmission of data, the nature of random noise in images is described here. Noise in video is usually defined as the unwanted components in each frame of a video stream [7, Chapter 4.5]. We wish to quantize these components and analyse their influence on each frame. Different noise sources may distort a frame with different types of noise. The random noise models used in this report is additive Gaussian noise, salt and pepper noise and multiplicative speckle noise.

Additive noise is described as shown in equation 3.4, where $f(\cdot)$ is the distorted image, $g(\cdot)$ is the original image and $q(\cdot)$ is the noise component [7, Chapter 4.5].

$$f(\cdot) = g(\cdot) + q(\cdot) \tag{3.4}$$

Gaussian noise is modelled using the equation above, where $q(\cdot)$ is described as shown in equation 3.5. Here, μ is the mean, while σ is the variance of the noise.

The Gaussian noise model is widely used, and is a very good model for thermal noise and, sometimes, film grain noise [7, Chapter 4].

$$p_q(x) = (2\pi)^{-\frac{1}{2}} e^{-(x-\mu)^2/2\sigma^2} \tag{3.5}$$

Salt and pepper noise can occur as a response to a range of events. Its characteristics is a few very noisy pixels in the image. The amount of noisy pixels, or the noise density, determines how badly distorted the image is. Salt and pepper noise can for example arise because of a bad digital link [7, Chapter 4.5] or because of faulty electronic components.

Multiplicative noise is another commonly used noise decomposition. Its description is shown in equation 3.6.

$$f(\cdot) = g(\cdot) \cdot q(\cdot) \tag{3.6}$$

Multiplicative speckle noise in images appear as a result of the actual capturing of the image, and is therefore often related to the camera itself.

Chapter 4

Equipment

The sections below describe the equipment used for developing the algorithms and for designing the FPGA circuits. Keep in mind that the VQA system are designed towards, but not tested on the described hardware.

4.1 Software for algorithm modelling and testing

MATLAB R2012b, together with the image processing toolbox was used for testing and developing the algorithms and metric data. This was also the software used in order to generate the resulting images and figures used in this report.

4.2 Software for implementing FPGA circuits

All circuit descriptions are written in VHDL, with the use of some external IP (Intellectual Property) components provided by Altera. Simulation of the HDL code was done using Aldec Active-HDL 9.2. Synopsys Synplify Pro version fpga_G201209SP1 was used for synthesizing the code during development, while the end results are provided by Altera Quartus II 11.1sp2 Web Edition.

4.3 Target hardware

The target platform of the system is an Altera Cyclone II FPGA [3] with version number EP2C70F896C6. The target FPGA contain 250 M4K blocks, the dedicated memory resources internal on the Cyclone II FPGA, which contain 4096 memory bits each. This equals to about 1 Mbit of internal storage that can be configured in various ways. The target FPGA also contain 150 dedicated multipliers that can be configured to support one 18×18 or two 9×9 multiplications. The target VGA circuit is an Analog Design AD9388A [24], which is a high quality, single-chip graphics digitizer.

Chapter 5

Presentation of the VQA algorithms

The following three chapters present and describe the design, modelling, implementation and verification of several VQA algorithms. The different algorithms are modelled and tested using MATLAB and implemented using VHDL code. Keep in mind that the design process that is depicted here is a naturally iterative process, meaning that the algorithms, the models and the code has been frequently changed in order to fit the specifications and in order to achieve the best results possible.

This chapter looks at how an FPGA can discover artifacts in a video stream and trace it back to board-level faults, such as the ones described in section 3.5 and in table 3.2. The chapter starts off by defining the system specifications by setting some specific boundaries. The next section looks at how one can utilize simple data value analyses in order to measure distortions. The latter sections look at specific algorithms for measuring the level of random noise and blurring of frames.

5.1 System specification

The VQA system inputs three 8-bit channels and explicit vertical and horizontal synchronization signals. There is also the pixel clock which is used to sample the pixel values. The systems have to support an input resolution of up to 1920×1080 pixels at 60 frames per second. This mandates a pixel clock frequency of at least 148.5 MHz [5, Chapter 4]. All models are to be implemented with a synchronous active high reset and a data enable signal for simple interfacing to the systems. Figure 5.1 shows the in- and outputs of the standard top module and table 5.1 includes a description of each signal.

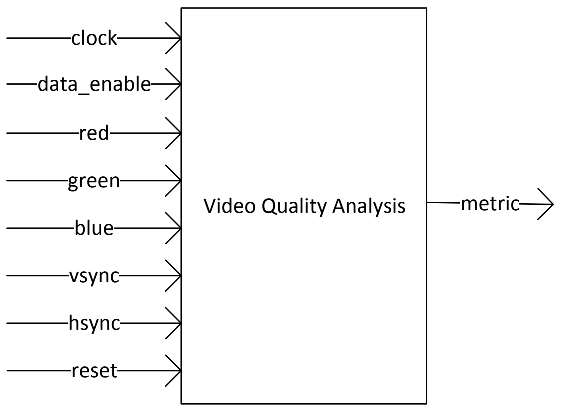


FIGURE 5.1: The top level module and its inputs and outputs.

TABLE 5.1: Description of the signals of the top level module.

Signal	Description
clock	Pixel clock.
data_enable	Signals that there is active data on the input.
red	Red channel. 8 bits.
green	Green channel. 8 bits.
blue	Blue channel. 8 bits.
vSync	Vertical sync signal. End of frame.
hSync	Horizontal sync signal. End of line.
reset	Synchronous active high reset signal.
metric	Output containing values describing the quality of the video.

Because of the inherent latencies provided with use of external memory, the system must only rely on internal block RAM for any buffering of the video signal. The systems will therefore only be able to buffer a few lines of an image at any time.

To keep the systems simple, multiple clock domains are undesired. The calculation of the metrics should therefore happen within the time it takes for one frame to arrive, and with the clock cycles available from the pixel clock.

It is desirable to be able to fit all VQA systems in one medium range FPGA. This means that the entire system should not use more than 70 kLEs and 1 Mbit of internal block RAM ($\approx$ 250 M4K RAM). However, the point of this assignment is not to create a fully functioning VQA system, but rather to examine and evaluate different possibilities for VQMs on an FPGA. Resource sharing is therefore not a requirement in this specification. One should also keep in mind that the specifications presented here are not all hard demands, but rather guidelines for the development of the metric algorithms. A summary of the resource specifications is presented in table 5.2.

TABLE 5.2: Summary of the system resource specifications.

Resource specification	**Maximum value**
Minimum system clock frequency	150 MHz.
Maximum system area	70 kLEs
Maximum block RAM usage	1 Mbit

It is important to ensure that the correlation between physical faults and the distortions are always sustained so that it is possible to both discover artifacts and to trace these back to the decisive faults. This will normally be upheld by keeping the processing of the color channels separate by avoiding transforms to other colour spaces.

5.2 Data Value Analyses

It is important to test whether the three color channels in a RGB video stream function as they should. In order to examine this, a module that analyses the basic properties of each color channel has been designed. The output of this system will consist of multiple metrics describing the maximum and minimum intensity of the red, blue and green channels and the occurrence of specific errors such as stuck-at faults. The different metrics for a single channel can be combined at the output into one metric for better overview. On the other hand, the metrics for the three channels should be treated separately throughout the analysis in order to enable the user to trace artifacts back to the correct physical faults.

The data value analysis is based on simple threshold testing of the incoming signal. The amounts of values that are higher or lower than a given threshold are logged, giving the possibility of creating a metric that describes the video systems ability to reach these values. If a well-designed test pattern is employed, this simple test system is believed to provide good data on the condition of the video stream hardware.

5.2.1 Reduced-reference assessment tools

It is possible to run a comparative check of the incoming stream of pixel values if some of the properties of the original input is known or can be predicted. The input can for example behave after a mathematical model, or have special properties such as a fully uniform histogram.

A test signal can be a static frame such as the one shown in figure 5.2. This signal is designed to verify that the values from the video transmission can reach its highest and lowest values, thereby validating the physical channels ability to pull the signal to the extremes. The quick change from low to high values also help in identifying other possible erroneous behaviour in the physical channels, such as the response time of the data transmission lines.

A similar test pattern that can be used to retrieve information on the video quality is another static image where the pixel values move from the lowest to the highest intensity values, as shown in figure 5.3. This linear signal distributes all values evenly and in a very predictable way.

By employing such a signal to the video input it is possible to test for both deviations in the linearity of the signal and loss of certain values. This signal pattern can also help in discovering noise, or even help in mapping the noise to a specific model. This can be done by comparing the values pixel values of the video stream to the mathematical model that describes the line in the plot in

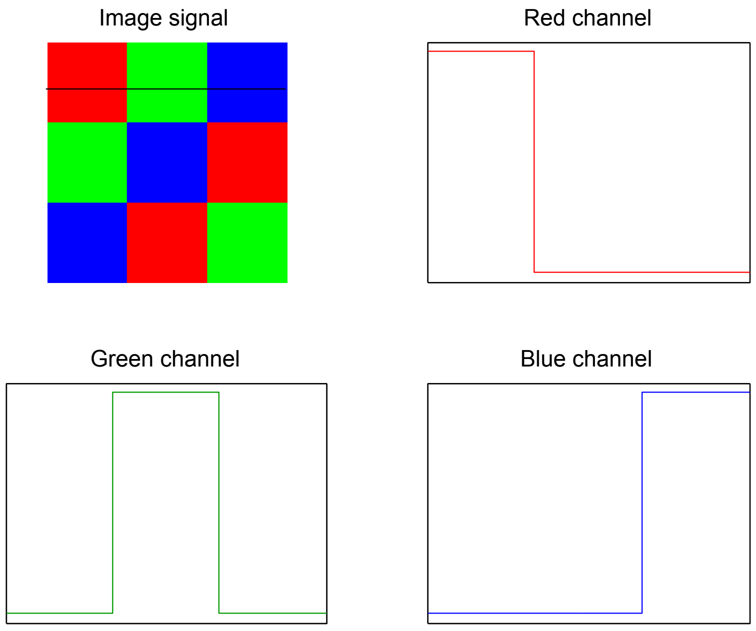


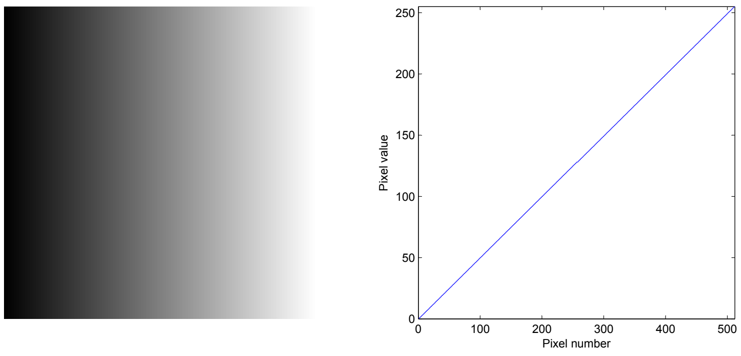
FIGURE 5.2: A possible version of a test signal that can help in revealing errors on the board-level of a circuit. The black line on the image signal representation shows which row is depicted for the red, green and blue channel in this setting.

figure 5.3. The deviations between the signal and the mathematical model poses as a good and very simple VQM.

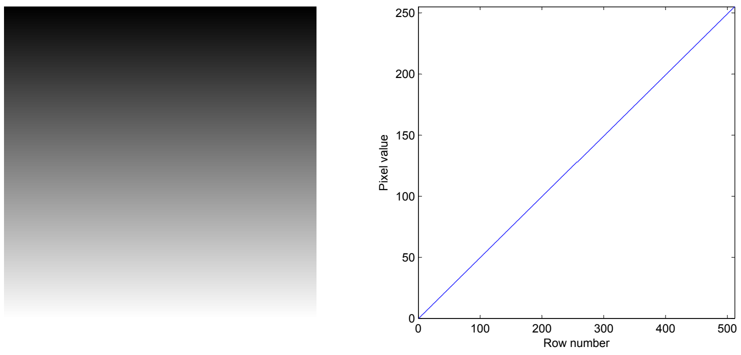
It is also important to verify that the horizontal and vertical synchronization signals are working properly. The test image depicted in figure 5.3a can be used to test for faults related to the horizontal sync by reading the values at the start and end of each line. If the values are not consistent with the test signal, there is an issue with the synchronization of the lines.

Similarly, the vertical sync signals the arrival of a new frame. It is not possible to utilize the signal in figure 5.3b in order to detect faults related to this, since all rows in this image are equal. However, by turning the image sideways, the same principle can be used for detecting video signals with poor vertical sync.

The frequency response of the video system can be tested by utilizing a test image similar to the one in figure 5.4 and figure 5.5. The first image consists



(A) Test signal used for discovering faults related to horizontal synchronization signals and a systems ability to create a linear signal.



(B) Test signal used for discovering faults related to vertical synchronization signals.

FIGURE 5.3: Two test images used for detecting synchronization errors and a video systems ability to produce a linear signal. The two signals are equal, but rotated. The plots show the signals when visualising the pixel values column-wise and row-wise, respectively.

of a sinus signal with increasing frequencies along the image width, allowing for measuring the frequency response of the system. The latter image is based on the same approach, but separates the different frequencies into different rows, allowing for an easier, but more time consuming analysis. Figure 5.6 shows plots of the different signals used for creating figure 5.5. The frequencies depicted in these images are only meant as descriptive examples, and are a lot lower then what should be used in an actual test scenario. In similar test equipment, the frequencies used range from 0.25 MHz to 5 MHz [20].

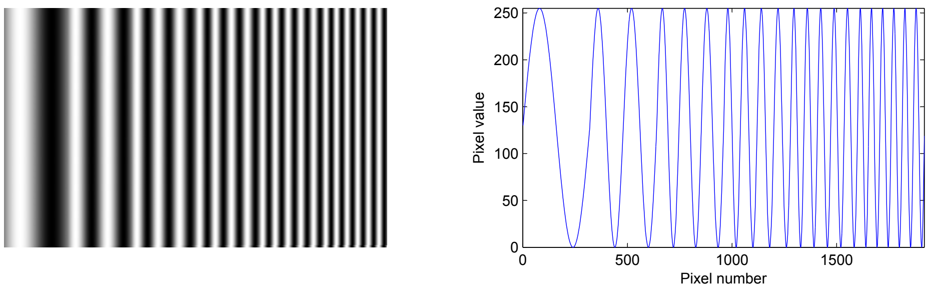


FIGURE 5.4: A test signal that helps in detecting inadequate bandwidth in video systems. The plot on the right shows the row data. All rows are the same.

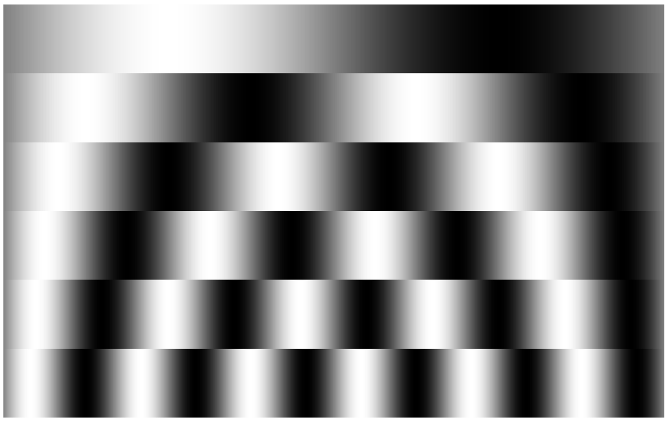


FIGURE 5.5: A test signal that helps in detecting inadequate bandwidth in video systems.

The test signals presented here can be used one at a time, or they can be combined into one frame containing sections of each test signal. In the latter case, the VQA systems must be specifically designed with a control unit for reading out the metric values at the correct times.

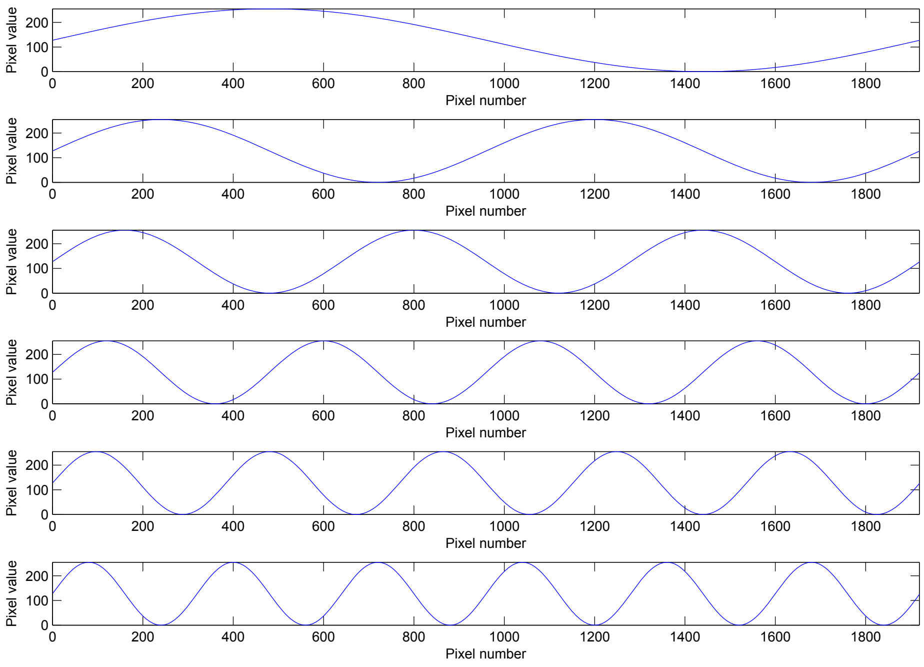


FIGURE 5.6: An example of the signals used for detecting inadequate bandwidth in video systems.

5.2.2 Testing for stuck-at faults

Figure 5.8 depicts the worst and best case of stuck-at faults when it comes to visual distortion. As the figure shows, depending on the significance of the faulty bit, it can be nearly impossible to spot stuck-at faults with the naked eye. An objective test for discovering the occurrence of such an error is therefore desirable. An analysis of such faults is possible by testing for the occurrence of both high and low representations of each channel bit, as long as the fault is not transient and is always active. However, if the error is somewhat transient, for instance because of a state machine or because of physical disturbances, then this test will require much greater complexity. A description of the intended functionality of the test system is shown in figure 5.7.

The system simply reads the input video stream from the specified channel and marks the registers if the different bit values have occurred. When the input consists of natural images, the testing may take a while since there is no way of

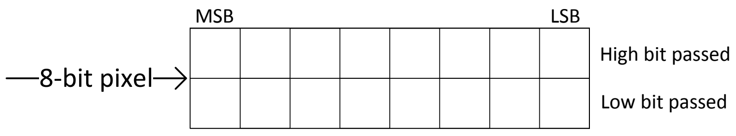


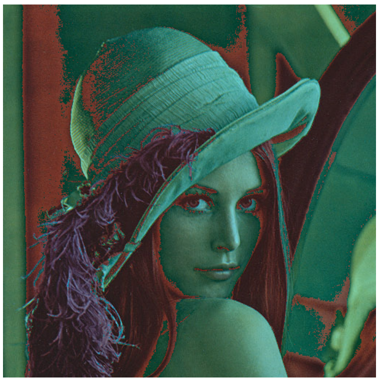
Figure 5.7: Eight bits are input to the assessment system. Depending on whether each bit is low or high, the appropriate cell is marked in the appropriate register. If one of the upper or lower cells fails to be marked, the respective location has a stuck-at-one or stuck-at-zero fault, respectively.

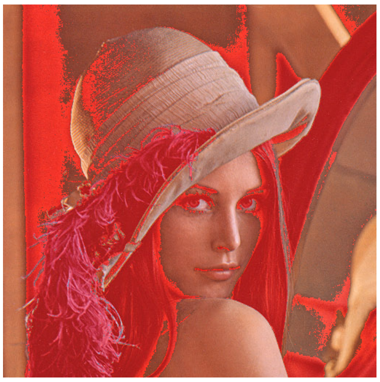
knowing weather all cases can be reached. However, if the test signal in figure 5.3 is employed, it is certain that all bit are set both high and low in just one frame. If the registers that are tracing the bit values are not all set to one, there is a stuck-at fault in the video system.

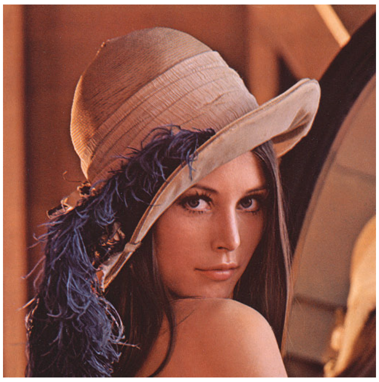
5.2.3 Histogram processing

Another extension to the data assessment algorithms is a histogram processing component. When studying histograms, one can determine if there are unnatural deviations by comparing it to some expected properties. For natural images, it is for example possible to look at the correlation between neighbouring bins. Sudden drops to zero and very high spikes at certain bins signifies that there may be faulty data because of for instance stuck-at faults.

The distribution of values may also point to faults in the system. The dynamic range of a frame signifies how well distributed the values are. Figure 5.9 shows a histogram plot of the Lena image. The dynamic range of the image is plotted as a line above the histogram. The covered values are all within the dynamic range of the image, however, the highest and lowest values are not covered. In order to test specifically for the system ability to reach the highest and lowest values, one can utilize test patterns such as the ones presented in figure 5.3.


(A) Image with the most significant bit stuck at zero.


(B) Image with the most significant bit stuck at one.


(C) Image with the least significant bit stuck at zero.

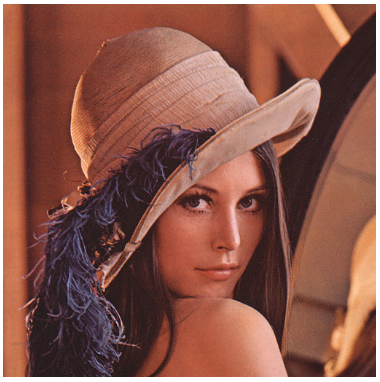

(D) Image with the least significant bit stuck at one.

FIGURE 5.8: Images showing that different stuck-at faults can result in very different visual outputs.

5.3 Random noise metrics

The amount of random noise in a video frame is assumed to correlate heavily with the level of oscillations in the data. To find the level of random noise in a video frame or a video stream, it is therefore possible to look at the intensity of these oscillations. To be able to compare the information in one frame to another, it is necessary to store some information. Keeping within the specifications of the system requires that relevant and significant information must be extracted from

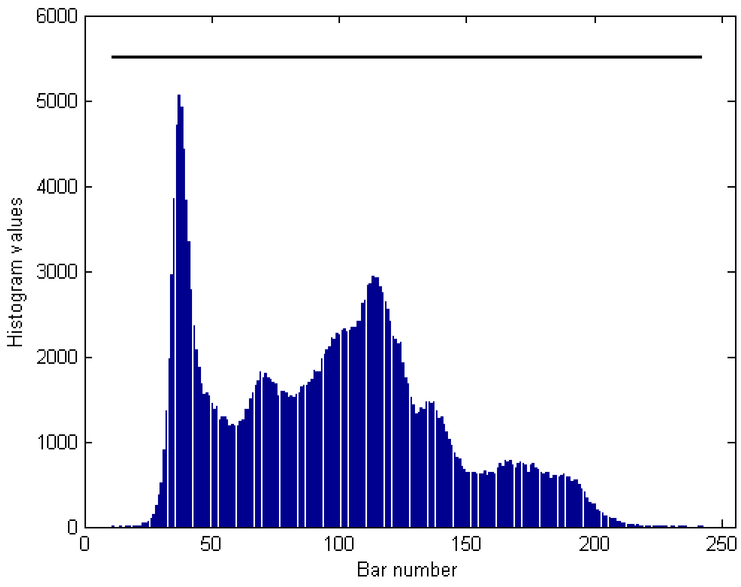


FIGURE 5.9: This figure contains a histogram plot showing the dynamic range of the grey scale Lena image. The blue line at the top indicates the full dynamic range of the image values.

the frame and stored efficiently, and then computed into understandable values that form the quality metric.

5.3.1 Accumulate and differentiate metric

A possible system that estimates the level of noise in a video stream is comprised by comparing the values of each row in adjacent frames. The values of a row are set according to the values of the pixels in the row, for instance by adding all values together. The values must then be stored in a register of a suitable size and compared to the value of the row of the next frame. The difference between the two values can then be used as a metric describing the level of noise in the video sequence.

This metric will discover unwanted oscillations in the intensity of the pixels in one row. The accumulation of the pixel values will function as a filter, removing random noise that oscillates at higher frequencies then the image frequency. Together with a metric that focuses on spatial noise in a single frame, this metric is believed to give good indications on whether a video stream is contaminated with any form of unwanted oscillations.

The metric may also provide information on horizontal lines caused by periodic and sporadic interference, since these lines often jump across the screen according to differences between the noise frequency and the video sampling frequency.

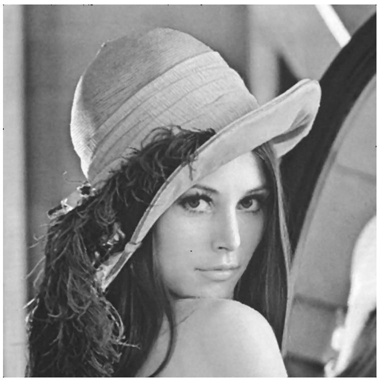
5.3.2 Median filtering metric

It is possible to use a median filter for discovering the level of noise in a single frame from a video stream. A median filter is a filter type that is heavily used in digital image processing for removing defective pixels in an image. The filter is especially effective when dealing with extreme outliers, such as when dealing with salt-and-pepper noise, but functions fairly well when working with other types of noise as well. If the output from this filter is subtracted from the original signal, one obtains an SAD which correlates with the amount of heavy changes in the spatial domain. This value therefore functions as a metric for the level of noise in a single video frame. Figure 5.10 demonstrates the principles of the system.

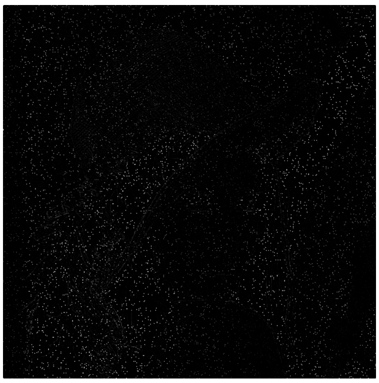
This metric will provide best results when utilized on images that are low on naturally occurring oscillations. However, it is believed that when looking at large, natural images and most artificial images, the resulting metric will correlate well with the intensity level of the distortion.

Median filters are great when working with salt and pepper noise, but for better results when trying to detect Gaussian noise and speckle noise, alternate filter types may be used. Wiener filters, bilateral filters and Frost filters have been used for detecting speckle earlier and may therefore prove to give more accurate results when such distortions are introduced to the video system. These filters are not further investigated in this thesis, and will be the result of further work.

(A) Lena image distorted with salt and pepper noise.



(B) Image in 5.10a sent through a median filter.



(C) The absolute differences between the distorted image and the filtered image.

FIGURE 5.10: Figures demonstrating the principles behind the median filter-based VQA system. It is believed that the sum of the pixels in figure 5.10c will function as a good metric for random noise.

5.4 Blur metrics

Figure 5.11 shows how blurring affects both the visual impression and the pixel values of an image. It shows that when an image loses some frequency components, it appears less sharp, or blurry. In order to determine weather the video stream has been blurred one can therefore look at the intensity of the edges in the image. It is also possible to look at the difference between a blurred version of the signal and the original signal, since a signal that consists of sharp edges will be more significantly altered by a blurring filter than a signal that contain smoother edges. Based on these properties, two approaches are proposed in the following sections.

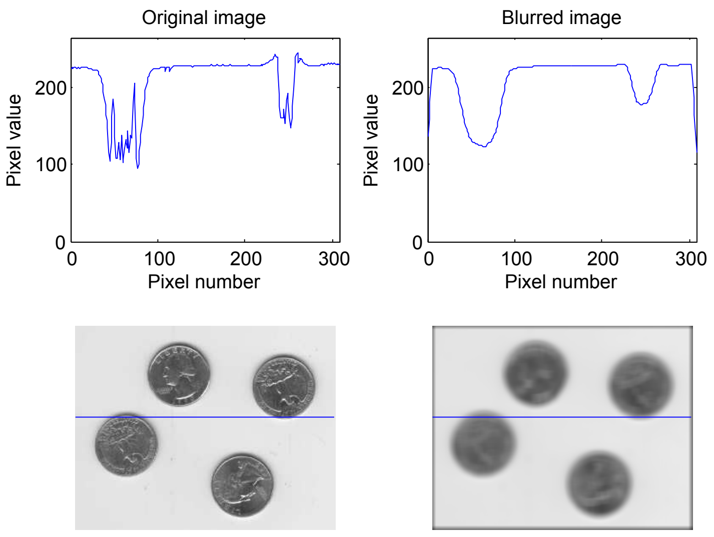


FIGURE 5.11: This figure shows two images with a highlighted blue row. The pixel values of this row are shown in the graphs above the images. It is easily noticed that the number of edges and the intensity of these are significantly lower in the blurred image.

5.4.1 Gaussian filtering metric

When an image is blurred because of a faulty electrical component, it loses some of its frequency components. Blurring an image using a Gaussian filter has a very similar effect, removing large changes between neighbouring data. An effect of this is that once the image has been blurred, it will be less affected by blurring the next time. This can be exploited in a VQA system by filtering the incoming image data using such a Gaussian filter and comparing the filtered image with the original one. The difference between the values signify whether the original image contains high frequency components.

Figure 5.12 shows the functionality of this approach by displaying the absolute differences between the original image and a blurred version of the image, in addition to the absolute differences between the blurred version and a version that has been blurred twice. As one can see, the plot from the first filtering contain far lower values than the plot from the second filtering.

A summation of the absolute differences should produce a metric that correlates well with the sharpness of an image or video frame.

5.4.2 Gradient metric

A more advanced way of analysing the sharpness of an image is performed by measuring the width of an edge [11]. This algorithm relies on an edge detection component for discovering the location of an edge, before it looks for the local extremes adjacent to all edge pixels. The length between the two local extremes provides a good measure for the blurriness of a video frame.

This method is, however, unnecessarily complex and sequential, requiring the filtering of the image, a threshold analysis, edge marking and hysteresis calculations in order to compute the metric. A better suited method is performed by utilizing the characteristics of gradients [14]. This is a much simpler and more efficient way of finding the hardness of an edge, since it only requires one filtering and a simple analysis of this result.

By filtering an image through a convolution kernel utilizing a Laplace operator one can find a good measure of the steepness and size of the edges in the signal. A disadvantage to this method is that this approach is very sensitive to heavy outliers, meaning that random salt and pepper noise will greatly affect the results. The algorithm is still believed to provide a very good metric in cooperation with other VQMs. The output from filtering the Lena image using a Laplace operator is shown in figure 5.13.

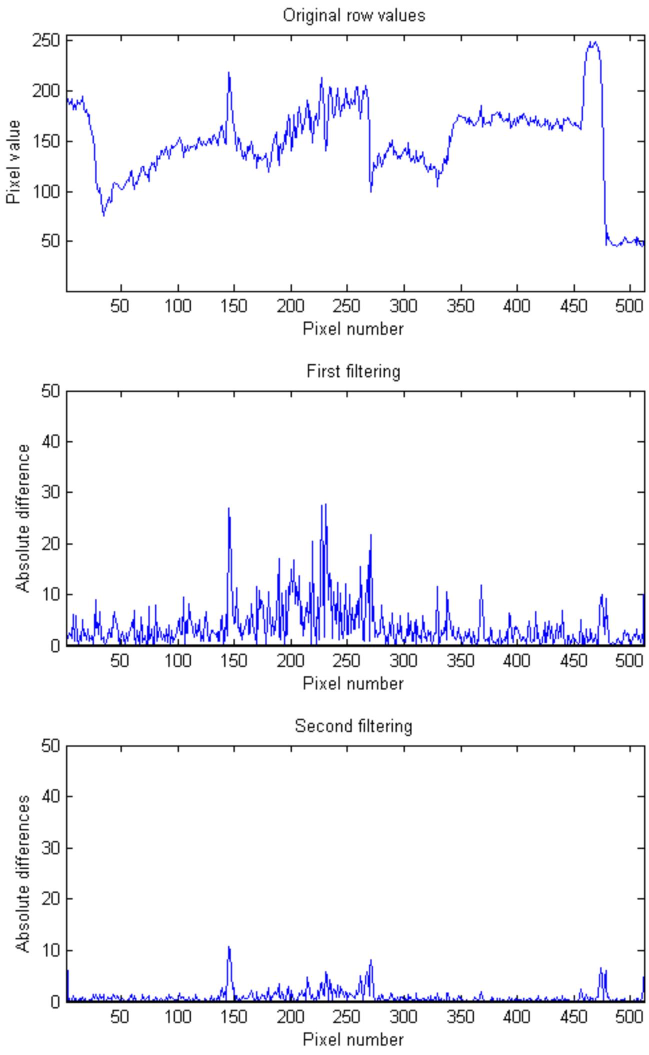


FIGURE 5.12: The upper plot shows the pixel values in one row of the Lena image. The middle plot shows the absolute differences between the original row and the row from the Gaussian-filtered version of the image. The lower plot shows the absolute differences between the rows in the filtered image and a version of the image that has been Gaussian-filtered twice.

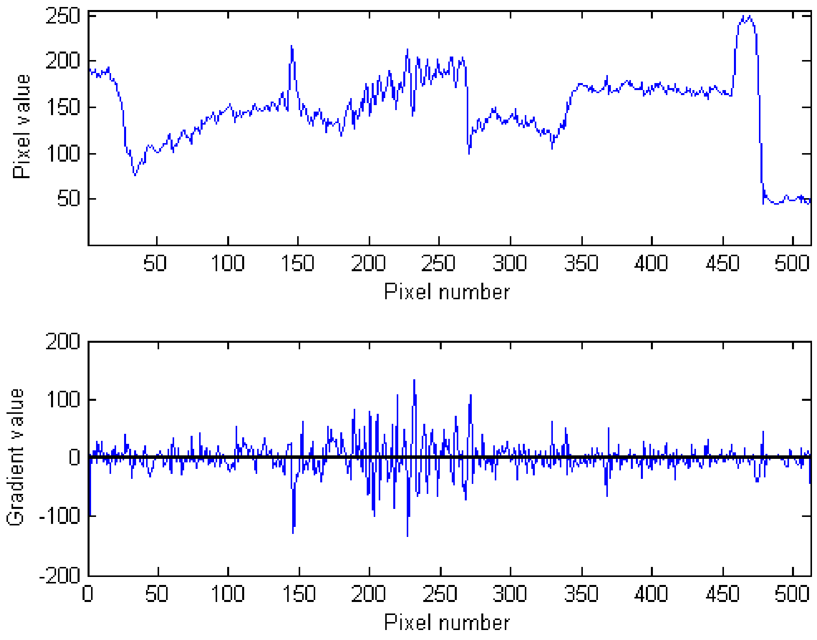


FIGURE 5.13: The upper plot shows the pixel values in one row of the Lena image. The lower plot shows the equivalent gradient values obtained from filtering with a Laplace operator.

5.5 Post processing

It is in most cases desirable to perform some form of post processing on the raw metric data from a VQA system. In many cases, the metric is naturally presented as an SAD (Sum of Absolute Differences). However, if the raw metric data has a very high dynamic range, it is often desirable to use an SSD (Sum of Squared Differences) to represent the metric. This will give a better representation when dealing with large differences. It is also usual to normalize the metrics in order to make the numbers more comprehensible.

Most video metrics are based on image metrics that are performed on each frame. It is however not possible to relate to all the frames in a video stream. Therefore, an averaging component is often part of the post processing. This component simply averages the value of the metric over a predetermined number of frames.

It may also be of interest to use this component for removing large outliers, such as black frames, from the data pool.

Chapter 6

Modelling and testing the algorithms

This chapter describes the modelling and testing of the algorithms. The models are designed to fit the specifications presented in chapter 5 and are tested according to the testing procedure presented below.

The first section of this chapter presents the design and execution of the different test benches. The general algorithm model is then presented, before the modelling and testing of each specific VQM is explained. The MATLAB model scripts are included in appendix A.

6.1 The tests and the criteria

To be able to test and evaluate the algorithms that are presented in chapter 5, a test bench involving the three natural still images presented in figure 6.1 has been developed. These images contain different levels of frequency components and different variations in scenery, and should therefore function well in the evaluation of the algorithms. Table 6.1 presents the size of the images and the dynamic range of the different color channels. To get a proper data pool for describing the performance of the metric systems, the images are contaminated with different distortions at different intensity levels. The correlation between the resulting metric and the level of distortion determines how well the algorithm performs. In addition to these results, the complexity of the algorithm and the difficulty of mapping the algorithms towards an FPGA is relevant to the metric evaluation.

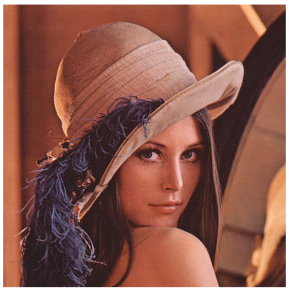
TABLE 6.1: The properties of the images used for testing the different VQA systems.

Metric test bench properties

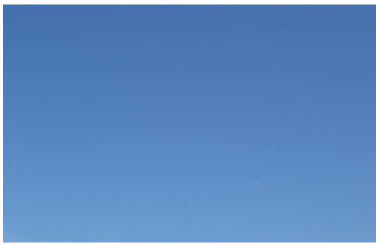
	Lena	Sky	Lake
Size	512×512	559×356	704×528
Dynamic range, red channel	$15 \rightarrow 255$	$63 \rightarrow 122$	$0 \rightarrow 255$
Dynamic range, green channel	$3 \rightarrow 238$	$109 \rightarrow 167$	$0 \rightarrow 255$
Dynamic range, blue channel	$0 \rightarrow 255$	$168 \rightarrow 211$	$0 \rightarrow 255$

The images are contaminated with four types of distortions, in addition to some specific artifacts for evaluation with some special cases and for testing with the data value analysis tools. The four general distortions are random additive Gaussian noise, salt and pepper noise, multiplicative speckle noise and blurring. The VQA systems are tested using all distortions at eleven incremental intensity levels, including the non-distorted versions of the images. This is to be able to evaluate both the sensitivity and the consistency of the different metrics. The commonly known PSNR metric is also calculated for all images with all of the distortions. Together with the visual impression of the images, this allows for better relations with the metric results.

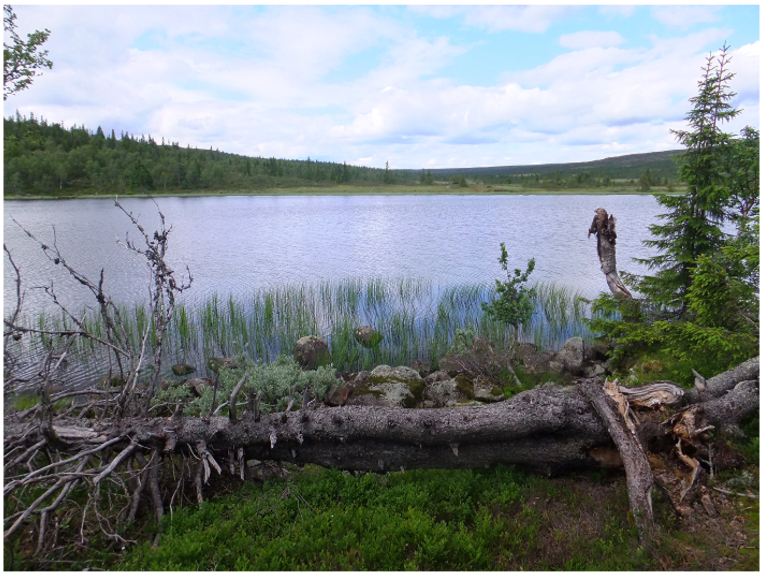
The random noise distortions are introduced using the function *imnoise()* from the MATLAB image processing toolbox, while the sharpness reduction is done by filtering the images using the function *fspecial()* as a Gaussian filter with a kernel size of 3×3 pixels. This filter is often referred to as a blurring or a smoothing filter. The mean of the speckle and Gaussian noise are set to zero, while the variances range from 0.01 to 0.1. The density of the salt and pepper noise also ranges from 0.01 to 0.1, while the variance of the Gaussian filter ranges from 0.3 to 1.0. The visual impairments of the distortions are shown in figure 6.2. These images contain distortions at the highest intensity levels available in the test bench.



(A) The famous Lena image containing both smooth areas and areas with high frequencies. The size of the image is 512×512 pixel. It is coded as an RGB bitmap using 8 bits per channel.

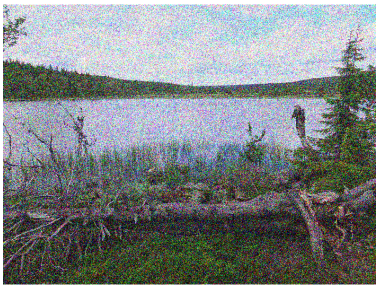


(B) An image of the blue sky with very small intensity changes. The size of the image is 559×356 pixels. It is coded as an RGB bitmap using 8 bits per channel.

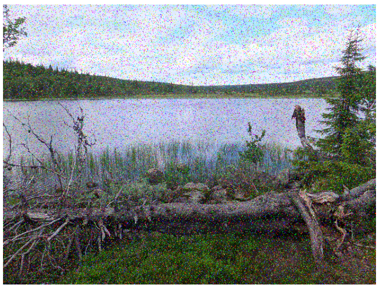


(C) An image of a lake with surrounding nature. There is a large amount of both small and large frequency changes all over the image. Notice especially the reeds, the water and the branches of the tree in the foreground. The size of the image is 868×614 pixels. It is coded as an RGB bitmap using 8 bits per channel.

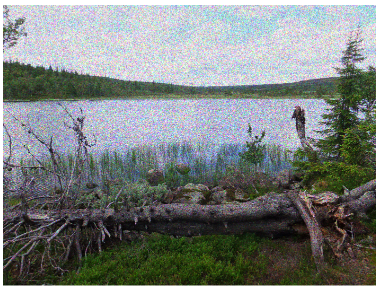
FIGURE 6.1: Different images used in the testing of the algorithms. The images have different frequency components and are very different with respect to both colors and the dynamic range of the image values.



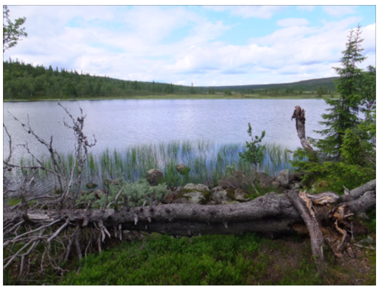
(A) Additive Gaussian noise.



(B) Salt and pepper noise.



(C) Multiplicative speckle noise.



(D) Blurring.

FIGURE 6.2: The *Lake* image contaminated with the different distortion types used in the testing. The distortions are of the highest intensity available in the test bench.

The test signals presented in section 5.2.1 are only used for testing some specific properties of certain metrics and are not used for testing in the same extent as the images presented in figure 6.1.

6.2 General algorithm modelling

The various no-reference assessment systems rely only on the video signal that is input to the system for computing the metric. This video signal is, as explained in chapter 3, a 24-bit RGB signal with external synchronization signals. The video signal can in theory represent anything, picturing any scenery and forming any abstract model. For the algorithms to be pure no-reference, they can not assume to know anything about this video signal. This requirement is not broken for any of the proposed VQA systems. It is, however, considered to be of great benefit to be able to compare the metric values with values calculated beforehand, or to values from previous tests. This is especially beneficial for the metrics that do not correlate well between image sceneries. For most metrics presented below, assumptions about video properties such as the frequency components of the input and the consistency of the scenery is believed to help greatly in their analysis.

To ease the use of memory, all analyses are done using image processing components that can be performed on small sections of the video frame. In order to keep within the system specification, filters utilizing data from more than two rows should not be used. This means that the largest available filter utilizes a 3×3 filter window. Since all of the metrics presented here must keep the color channels separate in order to not contaminate important data, there is a need for three processing components per metric; one for each color channel. This is taken into consideration in the modelling of all the algorithms below. Figure 6.3 shows a plot of a single color channel from an image row.

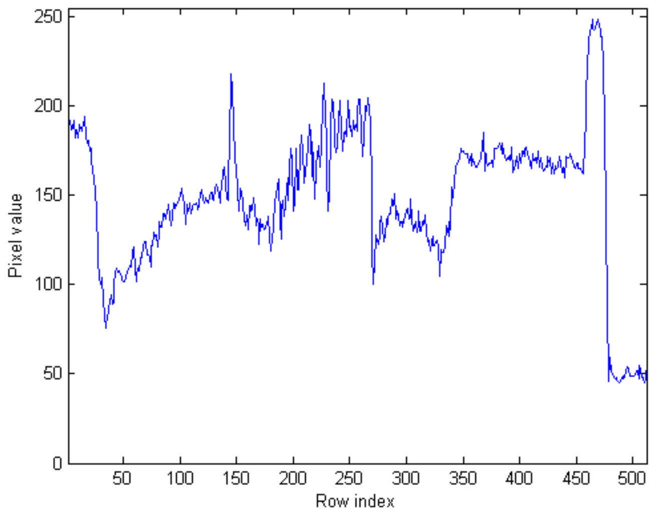


FIGURE 6.3: Plot of the red data in row 50 of the Lena image. The amount of internal memory made available by the system specification allows only for the buffering of two such rows per color channel.

6.3 Data value analyses

These simple quality assessment systems are tested and modelled individually using specific MATLAB scripts and information about the test signals presented in section 5.2.1. The systems include methods for analysing the frequency response, the highest and lowest pixel values, the systems ability to construct a linear signal, horizontal line flicker and faults in the synchronization signals.

6.3.1 Testing for stuck-at faults

The system is modelled as an iterative sequence, testing the state of all incoming bits. Whenever a specific bit state occurs, a variable is marked accordingly. One channel contains eight bits, meaning that there are 16 possible states. A minimum of eight states is always set, since the bits must have one of two values at any time.

The design of the module is relatively simple, meaning that no detailed tests or analyses are required. The model is tested using a small amount of natural images with stuck-at faults introduced. It is also tested with the images from figure 5.3, which covers all possible bit values.

6.3.2 Histogram processing

The histogram processing module is modelled and tested using the MATLAB image histogram function *imhist()*. Throughout these tests, the number of bins in the histogram is set to 256, which matches the number of possible pixel values.

The simplest test consists of checking the two extreme values, i.e. the highest and the lowest bins. A higher value than zero in these bins proves that the system can reach the extreme values. By reducing the number of bins, the model can be used to tests for groups instead of single values. Doing so lowers the complexity and the resource consumption of the component, but reduces the amount of information retrieved from the module. It will for example no longer be possible to test for the extremes, though this is easily complemented by the simple analyses presented in the prior sections.

The other test is a analysis on the dynamic range of the frames. This is found by iteratively searching for the first and last bin containing a value larger than zero. This is a demanding task on an FPGA, especially if there is a large number of bins. However, it can be combined with searching for bins containing zero, or bins that are very different from its neighbours. Such deviation in the histogram may point towards stuck-at faults or faulty sampling.

Overall, these analyses are hard to utilize without using the test signals, which allows for simple data comparisons. Using the test signal presented in figure 5.3 will for example produce a completely uniform histogram, where all bins contain the same specific value, while the signal presented in figure 5.2 will result in a histogram where only the lowest and highest bins contain a value above zero. This is verified in the testing of the tool.

6.4 Random noise metrics

The random noise metrics are tested primarily with images contaminated with random additive Gaussian noise. The metrics are also tested with images contaminated with salt and pepper noise and speckle noise to see how the resulting metric correlates with different random noise models. Lastly, they are tested using blurred images to see whether the metrics react to different levels of sharpness.

6.4.1 Accumulate and differentiate metric

For the testing of this algorithm, two images, representing two consecutive frames from a video stream, are used. The three color channels are separated and the algorithm is performed on each of them separately. The values of the channel pixels in the rows of both images are accumulated to form a sum representing the intensity of each row. The difference between the accumulated values from one frame, and the accumulated value from the other is calculated. The SAD from these calculations forms a value describing the changes between successive rows. By averaging this value over several frames, a metric describing the random noise in consecutive frames is formed.

The model is designed according to the algorithm shown below:

1. Load in the first image.
2. Accumulate the values of the pixels in the first row of the image and store the value.
3. Repeat point 2 for all rows in the image.
4. Load in the next frame.
5. Perform point 2 to 3 for this frame as well.
6. Find the absolute difference between the the values of the first row of the first image and the first row of the second image.
7. Repeat point 6 for all rows.
8. Find the sum of the absolute differences and store it as the metric value.

The algorithm has been tested using the test signals presented in section 6.1, and by using specific images containing horizontal line flickering. The line flickering is believed to affect the metric by creating large deviations in the accumulated values corresponding to the row where the flicker occurs. The behaviour of the algorithm when exposed to noise oscillating at frequencies lower than the image refresh rate, has not been tested.

6.4.2 Median filtering metric

This model analyses an image by filtering each color component through a two-dimensional median filter using the MATLAB *medfilt2()* function. The function uses a 3×3 filter window and sorts the nine inputs, before outputting the median value. This output is subtracted from the pixel element in the middle of the filter mask. The sum of the absolute values from this subtraction forms a sum of absolute differences (SAD).

An overview of the algorithm used for implementing the model is presented below:

1. Filter the image using a median filter.

2. Calculate the absolute differences between the input image channels and the result from the filtering.
3. Calculate the SAD and store it as the metric value.

The metric has been tested using the standard test bench presented above. It is believed that this metric will function well with all types of random noise, but that the best results will be achieved when testing with salt and pepper noise. Blurring is also believed to have a small, but present affect on the metric.

6.5 Blur metrics

The blur metrics are tested specifically using the images with reduced sharpness. They are also tested with the other distortions to see how they react on noise with different characteristics.

6.5.1 Gradient metric

The gradients in an image can be found by employing a convolution filter with the Laplacian kernel presented in 6.1. This model uses the function *conv2()* with the image and the Laplace operator as the input parameters. The different gradients are compared and the largest gradient forms the metric.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{6.1}$$

In addition to being tested with the images from the test bench, the metric is tested using the test signals from section 5.2.1. This is to determine how the gradients behave when affected by different frequency values.

The model is implemented according to the algorithm below.

1. Filter the image using a convolution filter with a Laplace convolution kernel.

2. Scan through the filtered image, comparing each gradient with the next. The largest value is stored in a register.
3. Store the largest gradient as the metric value.

6.5.2 Gaussian filtering metric

This metric system uses the same approach as the system presented in section 6.4.2, where the input image is filtered and the sum of absolute differences is calculated using the input and the results from the filtering.

The filter behaviour is implemented using a convolution filter and the kernel presented in equation 6.2. The MATLAB model uses the function *conv2()* with the image and this kernel as its parameters.

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \times \frac{1}{16} \tag{6.2}$$

The model is tested using the standard test bench. The metric system is believed to produce good results, especially when looking at images with few natural high frequency components. It is believed to have great robustness towards other distortions.

The model is implemented according to the algorithm below.

1. Filter the image using a convolution filter with the Gaussian convolution kernel.
2. Calculate the absolute difference between the original input and the result from the filtering.
3. Find the sum of the absolute differences.
4. At the end of the frame, store the sum of absolute differences as the metric value.

Chapter 7

Implementation and verification

The implementation and verification of the algorithms are presented in this chapter. All implementation is done using VHDL and simulated using Active-HDL with a range of test benches. There has been a focus on making the modules as generic as possible in order to allow for heavy reuse of the components.

The estimated clock frequency and the amount of LEs used are the main concerns of the implementation of the algorithms. Apart from certain tweaking of the system in order to meet these concerns, the systems are implemented so that they closely resemble the models provided in chapter 6.

The different VQA systems are implemented separately, allowing for individual verification and testing of the metrics. This means that there are five separate systems; one for each of the specially developed NRVQA systems and one for the data value assessment system. Certain issues such as resource sharing have not been dealt with here, as they are not part of the assignment. The issues are however commented in the respective sections and thoroughly discussed in chapter 9.

The first section describes the implementation of the most frequently used components. The successive texts describe how all VQA systems are implemented in detail, and the last section describes how the general verification of the VHDL implementations has been done.

7.1 Frequently used components and values

Some important generics recur in several component entities. These values represent the size of the frame, the bit width of the pixel data and the number of samples used in the averaging of the metric values. All modules are implemented so that they should function with most values, but are only tested and verified using a frame size of 1920×1080, a data width of 8 bits and a sample number of 32 for metric averaging.

Two components occur in several of the VQA systems. These are the row buffer and the averaging component. The interfacing to these modules and their implementation is explained in the following sections.

7.1.1 Row buffer

The row buffer is a component that is needed whenever a frame is to be spatially filtered using a filter window. This includes both median filtering and convolution filtering. The implementation of the row buffer is done according to the architecture described in section 3.3.1.

The module has the inputs and outputs as presented in table 7.1. Since the systems developed here only utilize a 3×3 filter mask, there has been made no effort in making the filter mask size generic. The module is, however, generic with respect to the channel width and the size of the frame.

The *data out* channels present the data corresponding to three consecutive rows. As the data is fed into the component, the internal buffers are filled up, and after a start up latency equal to twice the number of columns in the frame, the correct row values appear at the output. The architecture of the buffer is shown in figure 7.1.

As shown in the figure, when utilizing this module with a convolution filter, the filter window must consist of three shift registers of three cells each. The

TABLE 7.1: Description of the input and output signals of the row buffer module.

Row buffer interface description	
Generics	
channel_width	The width of the data input channel as an integer. The default value is 8.
rows	The number of rows as an integer. The default value is 1080.
cols	The number of columns as an integer. The default value is 1920.
Inputs	
clk	Pixel clock.
reset	Synchronous active high reset signal.
data_enable	Signals when the data input is active.
data_in	The data values represented as a natural number with a generic range depending on the channel width.
Outputs	
data_out_1	The data corresponding to the first row represented as a natural number.
data_out_2	The data corresponding to the second row represented as a natural number.
data_out_3	The data corresponding to the third row represented as a natural number.

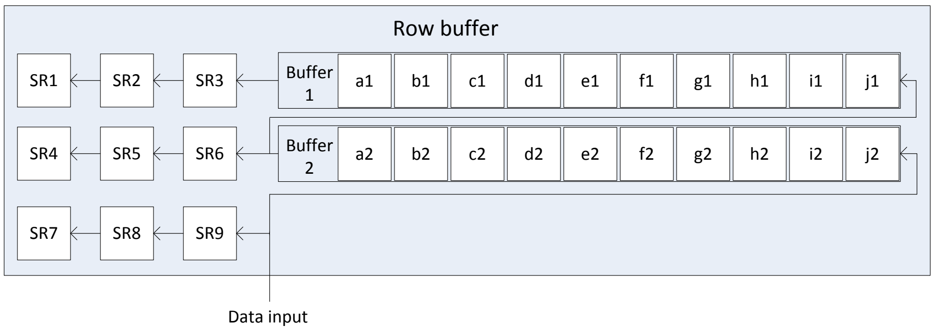


FIGURE 7.1: Architecture of the row buffer module.

outputs of these registers make the input to the filter computations. The median filter does not require these additional registers, as they are included in the filter computation architecture.

The internal buffers are implemented as one component using Alteras RAM-based shift register IP component. The component is called ALT_SHIFT_TAPS and contains some additional features not found in a conventional shift register [25]. The component is part of the *altera_mf* library. Specifying the number of taps in the register allows for data to be read out at specific points. By utilizing this, the desired behaviour of the row buffer is easily achieved. The configuration of the shift register is shown in listing 7.1.

```
ALTSHIFT_TAPS_component : ALTSHIFT_TAPS
GENERIC MAP (
  intended_device_family => "Cyclone II",
  lpm_hint => "RAM_BLOCK_TYPE=M4K",
  lpm_type => "altshift_taps",
  number_of_taps => 2,
  tap_distance => cols,
  width => channel_width
)
PORT MAP (
  aclr => reset,
  clock => clk,
  clken => clken,
  shiftin => data_to_ram,
  shiftout => shiftout,
  taps => data_from_ram
);
```

Listing 7.1: Configuration of the shift register.

7.1.2 Averaging

Since the metrics are based on analyses of single frames in the video stream, a component that averages the metric over several frames is needed in order to be able to relate to the metric data. This averaging component is constructed as shown in figure 7.2, using an n-bit shift register, where $n-1$ is the number of averaging samples, and circuits for adding and subtracting the first and last values in the register, respectively. The sum resulting from this is divided by the number of averaging elements, and the result is presented at the output. It is important that the number of averaging elements is set to a power of two in order to save resources when doing the division. No block-RAM is explicitly used in the implementation of this module.

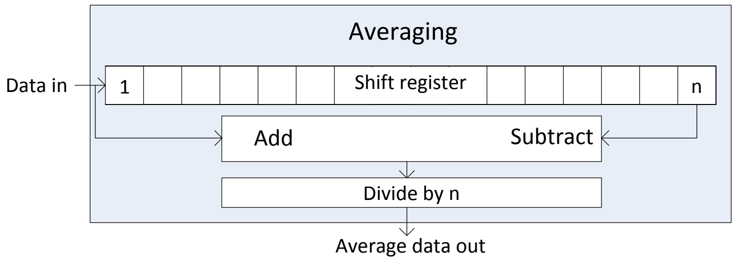


FIGURE 7.2: Description of the implementation of the averaging register.

7.2 Data value analyses

The simplest data value analysis is implemented by logging the occurrence of the extremes in each data channel. The number of such occurrences is stored in a register, which forms a metric correlated to possible faults in the video system.

The occurrence of simple stuck-at faults is logged using the specific stuck-at fault testing module, which is presented below. The tests related to the distribution of the data values are implemented using the histogram processing component.

7.2.1 Testing for stuck-at faults

The stuck-at fault analysis module is designed as a component in the data value analysis system, but can also function by itself, providing information on a data channels ability to reach both high and low values. The module is generic with respect to the channel width, but is only tested and verified using 8-bit data values. An overview of the inputs and outputs of the module is shown in table 7.2.

This simple module consists of two 8-bit registers, one for high and one for low bits, for storing whether the bit values has been covered or not. A low bit signifies

Table 7.2: Description of the signals of the stuck-at fault testing module.

Stuck-at analysis system interface description	
Generics	
channel_width	The width of the data input channel as an integer. Default value is 8.
Inputs	
clk	Pixel clock.
reset	Synchronous active high reset signal.
data_enable	Signals when the data input is active.
data_in	Generic length std_logic_vector. Default length is 8 bits.
Outputs	
stuck_at_one	Generic length std_logic_vector. Default length is 8 bits.
stuck_at_zero	Generic length std_logic_vector. Default length is 8 bits.

that the bit values have not yet been proven to occur, while a high value means that the values have occurred.

The registers are set to zero at reset and a process sets the registers according to the input data channel. A decoder generates a value based on these registers, creating a metric that can be used directly in order to establish the probability of stuck-at faults in the video system.

7.2.2 Histogram processing

In order to be able to process the histogram, the data must first be gathered from the incoming frames. This is done by accumulating the counts for each pixel value. For simplicity, this histogram accumulator is implemented using an array of counters enabled by the different data values, as described in section 3.3.3. Although the decoding of the data requires a lot of expensive logic, this was considered to be the best approach compared to an implementation using dual-port memory.

An overview of the interface to the histogram accumulator is shown in table 7.3. The module is generic with respect to the input data channel width, the number of bins and the size of the registers containing the accumulated values.

Figure 7.3 describes the implementation of the histogram accumulator. The counter array is an array of counter components with enable signals coming from

TABLE 7.3: Description of the signals of the histogram accumulator module.

Histogram accumulator interface description	
Generics	
channel_width	The width of the data input channel as an integer. The default value is 8.
bin_max_value	The max value of the registers as an integer. The default value is 4095.
number_of_bins	The number of bins as an integer. The default value is 256.
Inputs	
clk	Pixel clock.
reset	Synchronous active high reset signal.
index	Natural integer for choosing the which counter value to output. It has a generic range from 0 to number_of_bins.
data_enable	Signals when the data input is active.
data_in	Natural integer connected to a decoder for enabling the accumulation of a single counter. Generic size set by channel_width.
Outputs	
count	Natural integer with generic range from 0 to bin_max_value.

the decoder. The decoder makes sure that only one of the counters is enabled at one time. A multiplexer is connected at the output of the counter array, allowing for the readout of one specific bin. The counters used in the counter array are generic with respect to the register size and the channel width. They increment the register values if the clock enable signal is high at the rising edge of the clock.

The greatest issue of this component is the readout and analysis of the histogram values. An analysis of the data to form a metric is complex and requires multiple clock cycles. This is solved most easily by pipelining the process, utilizing the arrival of the next frame for going through all of the bins. Since this method relies on the use of a lot of resources, it is not of great interest for this thesis. It is is therefore not discussed further here, and will instead be the product of further work.

7.3 Accumulate and differentiate metric

The inputs and outputs of the top module is presented in table 7.4.

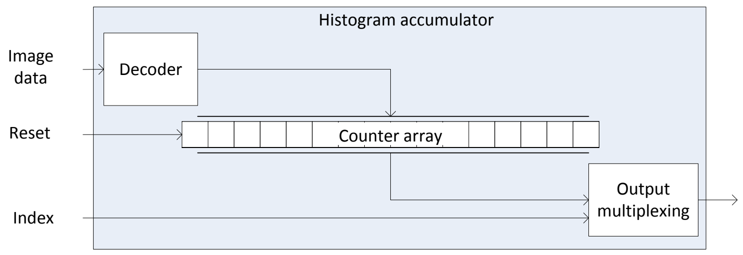


FIGURE 7.3: Description of the implementation of the histogram accumulator.

TABLE 7.4: Description of the signals of the accumulate and differentiate module.

A & D interface description	
Generics	
channel_width	The width of the data input channel as an integer. The default value is 8.
rows	The number of rows as an integer. The default value is 1080.
cols	The number of columns as an integer. The default value is 1920.
Inputs	
clk	Pixel clock.
reset	Synchronous active high reset signal.
data_enable	Signals when the data input is active.
vSync	Control signal for signalling the EOF.
hSync	Control signal for signalling the EOL.
red	The red channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
green	The green channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
blue	The blue channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
Outputs	
data_out	Metric data represented as a natural integer.

The system is implemented using an accumulator, a shift register and an SAD component, as shown in figure 7.4. The shift register has a generic length and width in order to function with different frames and data inputs. The control module is not shown in the figure in order to simplify the system overview.

The accumulator adds the values of a single row together, forming a sum of image

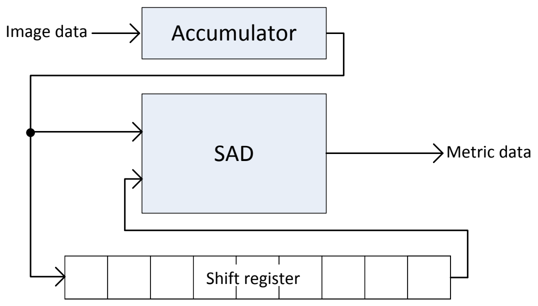


FIGURE 7.4: Realization of system for detecting random noise using row comparison.

values. At the end of each line, the output from the accumulator is fed into the SAD component and into the shift register. When the whole frame worth of rows are accumulated, the values of the first row is at the output of the shift register. The SAD component calculates the absolute differences between the output from the shift register and the output from the accumulator, which now outputs the accumulated values of the first row of the next frame. The SAD component outputs the raw metric as the sum of the absolute differences between the row values of two neighbouring frames.

The shifting of the values in the shift register is enabled and the accumulator is reset at the rising edge of the horizontal sync signal. Likewise, the SAD module and the accumulator is reset at the rising edge of the vertical sync signal.

7.4 Median filtering metric

This VQA system has the inputs and outputs as defined in table 7.5, and is generic with respect to the size of the frames and the width of the data channel.

Figure 7.6 shows the architecture of the median filter component. The internal architecture is based on the one presented in section 3.3.2. The filter can be used with frames of arbitrary sizes and with arbitrary data bit widths.

The median filter component provides both the filtered data and the original data to the upper module, so that the metric can be created by comparing the two

TABLE 7.5: Description of the input and output signals of the median metric VQA module.

	Median metric interface description
Generics	
channel_width	The width of the data input channel as an integer. The default value is 8.
rows	The number of rows as an integer. The default value is 1080.
cols	The number of columns as an integer. The default value is 1920.
Inputs	
clk	Pixel clock.
reset	Synchronous active high reset signal.
data_enable	Signals when the data input is active.
vsync	Control signal for signalling the EOF.
hsync	Control signal for signalling the EOL.
red	The red channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
green	The green channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
blue	The blue channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
Outputs	
metric_red	The metric for the red channel represented as a natural integer.
metric_green	The metric for the green channel represented as a natural integer.
metric_blue	The metric for the blue channel represented as a natural integer.

values. The original data is provided by a direct routing from the filter window. Pipelines are added between the components in order to obtain a large enough maximum frequency.

7.5 Gradient metric

This VQA system has the interface described in table 7.6. It is generic with respect to the size of the frames and the width of the data.

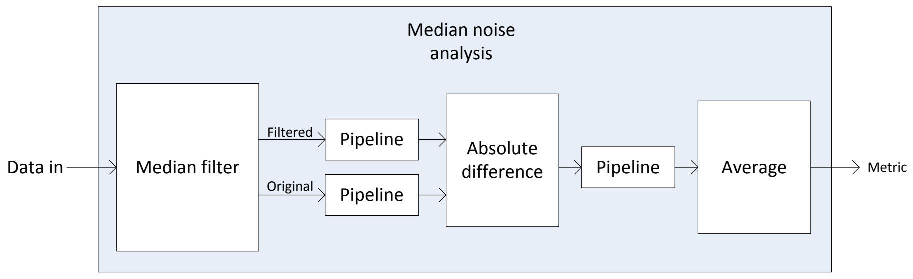


FIGURE 7.5: Description of the implementation of the single channel median noise metric architecture.

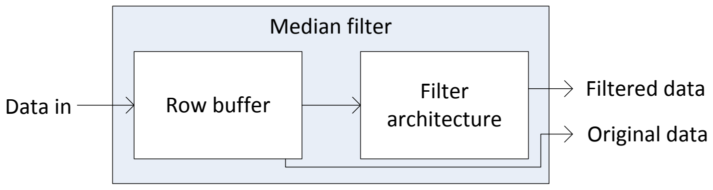


FIGURE 7.6: Description of the implementation of the median filter architecture.

The inner components of the module is shown in figure 7.7. A pipeline has been introduced after the convolution filter in order to increase the max frequency to a value above the minimum pixel clock frequency. The convolution filter outputs a value between -1020 and 1020, which are the lowest and highest numbers possible using the Laplace operator. Since only a positive value is needed, the absolute value is calculated before a comparator circuit compares the filtered value with the prior values from the current frame. If the new value is larger, it is replaced with the prior value. When all values have been compared, the result is sent to an averaging component. This component can be omitted, depending on the use of the metric.

The convolution filter is employed in order to compute the gradients of the frames. This component uses the same row buffer component as used in the median filter module, except from a few modifications since the input signal is not needed for the computation of this metric. The architecture of the convolution filter component is shown in figure 7.8. The filter component inputs the filter parameters as generics. The default values are the parameters for the Laplace operator, which

TABLE 7.6: Description of the input and output signals of the VQA module using gradients.

Gradient metric interface description	
Generics	
channel_width	The width of the data input channel as an integer. The default value is 8.
rows	The number of rows as an integer. The default value is 1200.
cols	The number of columns as an integer. The default value is 1920.
Inputs	
clk	Clock signal.
reset	Synchronous active high reset signal.
data_enable	Signals when the data input is active.
vsync	Control signal for signalling the EOF.
hsync	Control signal for signalling the EOL.
red	The red channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
green	The green channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
blue	The blue channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
Outputs	
metric_red	The metric for the red channel represented as an integer with a range from 0 to 1020.
metric_green	The metric for the green channel represented as an integer with a range from 0 to 1020.
metric_blue	The metric for the blue channel represented as an integer with a range from 0 to 1020.

is the operator that is used in this VQA system. This component includes three 1×3 shift register. These are the cells forming the filter window to the left in figure 7.1.

The border values are zero-padded by setting all non-defined values to zero. The metric is not averaged, since its function is to prove that the video transmission systems response time is fast enough. An averaging would compromise this information.

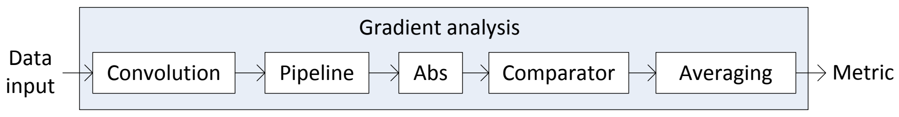


FIGURE 7.7: Overview of the components in the blur estimation module that computes the gradients in order to analyse the sharpness of the frames in the video stream.

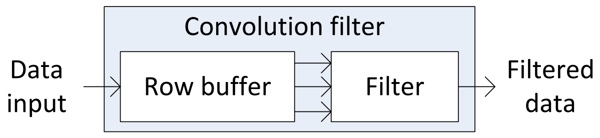


FIGURE 7.8: Overview of the architecture of the convolution filter.

7.6 Gaussian filtering metric

The Gaussian filter is implemented in the same way as the Laplacian filter, utilizing convolution using a row buffer and a filter calculation component. However, to apply the division by sixteen, a shifting by four places is performed after the filter computations. An extra pipelining register was inserted to keep the maxiumum frequency above the required 150 MHz. The computational analysis of the filter results are done in the same way as with the median metric. That is, the results from the filtering is compared to the original data and outputted as a sum of absolute differences. The same pipelines are used, and there is an averaging of the metric over a number of frames.

The interface to the module is described in table 7.7

7.7 Verification

The implemented components have been verified through simulation using specially developed test benches. The test vectors utilized by these test benches are generated using MATLAB, by printing the values of test images to a text file. The test vectors derive from both artificial and natural images of the sizes 512×512 and 1920×1080. The text file is read into the VHDL test benches

Table 7.7: Description of the input and output signals of the VQA module utilizing a Gaussian filter.

Gradient metric interface description	
Generics	
channel_width	The width of the data input channel as an integer. The default value is 8.
rows	The number of rows as an integer. The default value is 1200.
cols	The number of columns as an integer. The default value is 1920.
Inputs	
clk	Pixel clock.
reset	Synchronous active high reset signal.
data_enable	Signals when the data input is active.
vsync	Control signal for signalling the EOF.
hsync	Control signal for signalling the EOL.
red	The red channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
green	The green channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
blue	The blue channel pixel values represented as a natural integer with a generic range from 0 to $2^{channel\_width} - 1$.
Outputs	
metric_red	The metric for the red channel represented as a natural integer.
metric_green	The metric for the red channel represented as a natural integer.
metric_blue	The metric for the red channel represented as a natural integer.

and used as input to the different system components. The resulting values are compared to the values from the MATLAB models, verifying that the VHDL modules work as intended.

In addition to verification by simulation, the systems are verified by analysing the synthesis results. These analyses consists of verifying that the a plausible amount of resources are used and that the correct technologies are synthesized.

Chapter 8

Results

Several different video quality metric systems and video assessment tools have been modelled and tested in MATLAB. The key components in these systems have been implemented using VHDL code and synthesized for an Altera Cyclone II FPGA using Altera's software tools. The testing of the algorithms establishes whether the metrics correlate sufficiently with different types of distortions and if the assessment tools function as intended. The synthesizing of the components ensures that the algorithms can be mapped to an FPGA, and that they work within the given specifications.

The goal of this assignment is to provide a range of quality assessment algorithms and analysis tools that can run on a single FPGA, providing information on the state of the video system hardware. This chapter presents results from the testing of these tools and the results from the implementation and synthesizing of the VHDL models of the systems.

All results regarding the metric algorithms are provided from tests performed using MATLAB, meaning that this report does not present results from any working quality metric systems running on a physical FPGA. The behaviour of the VHDL models are however validated by simulation and by comparing the MATLAB model results with the VHDL model results.

Section 8.1 present the most important results from the algorithm analysis done in MATLAB, while section 8.2 presents the results from designing, implementing and synthesising the prototype VQA systems, and some important sub components specifically.

8.1 Algorithm test results

The algorithms were tested as described in chapter 6. The results from the most important tests are rendered here, together with some illustrations showing the key properties of the metrics. The results are presented to show how well the different metrics correlate with the intensity of the specific distortions, and if the metrics are heavily affected by other types of distortions. The data from both the red, green and blue color channels are presented in each plot, marked with their respective colours. Numerical results from all tests can be found in appendix B.

The images can be compared to the data from PSNR calculations. Figure 8.1 shows the resulting data from this metric when utilized on the image *Lake* with different distortions at different intensities. It is apparent that the distortions caused by the smoothing of the image are small at first, but significantly increasing until it levels out at the latter samples. The distortions caused by the random noise causes the PSNR to drop in a far more linear fashion.

8.1.1 Data assessment tools

The results from testing the various data assessment tools show that they work as intended, providing information on key image data characteristics.

The histogram provides the data needed for calculating the dynamic range of the images, giving the desired information about the highest and lowest pixel values. It is worth noticing that many distortions does affect the histogram analysis, but that these affects are neglected and not processed further because of the inconsistency of these results.

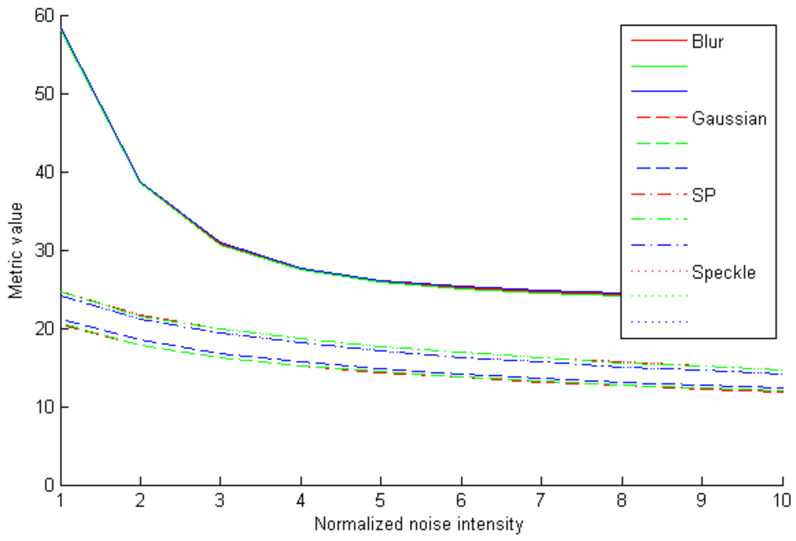


FIGURE 8.1: The plots shows the PSNR data when this metric is calculated using the *Lake* image with different distortions at different intensities.

The simple model designed for discovering stuck-at faults works as predicted. Images that are contaminated with such artifacts, even those with no clear visual distortions are successfully marked with the appropriate fault.

8.1.2 Accumulate and differentiate metric

When the system is utilized on static video where all frames are identical, i.e. contain no noise, this metric is equal to zero. Results from testing with the test bench show that when this metric is utilized on frames that contain Gaussian, salt and pepper and speckle noise, the metric increases steadily as the intensity of the noise increases. This behaviour is shown in figure 8.2. The system does not respond to blurring of the signal, since this does not imply any random oscillations. A constantly blurred video signal therefore results in a metric value equal to zero.

As figure 8.2 shows, the metric increases steadily when the intensity of the random noise grows more intense. This proves the functionality of the metric with respect to spatial noise in single frames. The metric is consistent when utilized on low intensity distorted frames of different sceneries, but is less consistent when used on the highly distorted versions. For example, a highly distorted version of the *Sky* image gives a metric value of 574 665, while the *Lake* image with the same distortion intensity gives the value 961 209.

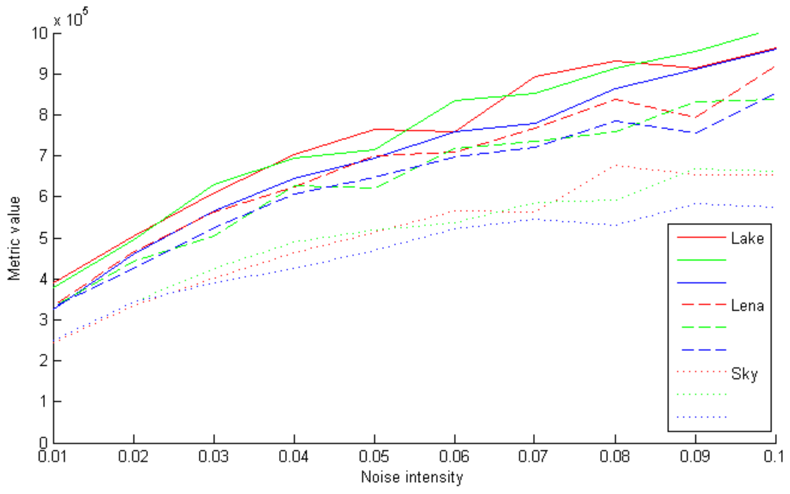


FIGURE 8.2: This figure depicts the metric output presented in table B.5. The x-axis is the variance of the noise added to the image, while the y-axis is the output from the metric.

When frames distorted by horizontal line flickering is input to the system, the metric itself reacts poorly, increasing the value only slightly. However, a significant change can be seen if the difference of each row value is analysed.

The metric value is very consistent when comparing the different color components of the images. The values varies only by a fraction between the color channels.

8.1.3 Median filtering metric

A seen in table 8.1, the median filter metric is far less consistent when it comes to changes in scenery, when compared to the A & D metric. The results from testing with the different images show that the images which contain far less high frequency components than the others results in a far lower metric value.

TABLE 8.1: Results from testing the median metric with the original test bench images.

Original images			
Channel	Red	Green	Blue
Lena	912 663	823 329	929 216
Sky	167 827	101 753	124 613
Lake	3 471 989	3 522 159	3 233 420

The metric values increase as the intensity of the distortions increases for all random noise. The amount of which they increase is very consistent, varying only by a tiny fraction. For example, the changes in metric values from images with Gaussian noise variance of 0.1 to 0.2 are $1.7 \cdot 10^{10}$, $1.4 \cdot 10^{10}$ and $1.8 \cdot 10^{10}$ for the Lena, Sky and Lake images, respectively.

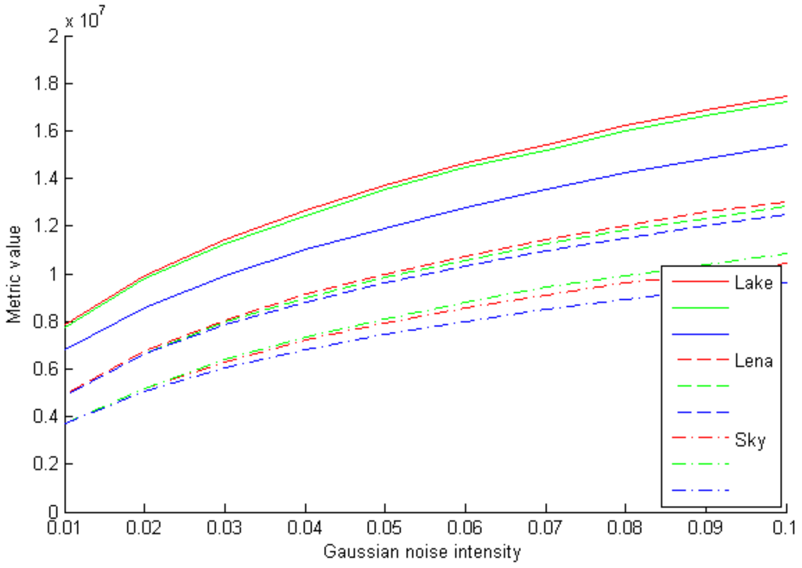


FIGURE 8.3: Results from testing the Median metric with the images from the test bench contaminated with different levels of Gaussian noise.

When tested with images exposed to frequency loss, the metric value is continuously reduced with an increasing level of blurring, implying that the algorithm reacts to blurring as well as the types of random noise.

8.1.4 Gradient metric

The gradient metric values resulting from calculations using the original test bench images are shown in table 8.2. There is little consistency between the metrics from the different images, in addition to less consistency between color channels, when compared to the other metrics.

TABLE 8.2: Results from the median metric when the original images (no introduced distortions) are used for testing.

Original images			
Channel	Red	Green	Blue
Lena	378	347	415
Sky	237	328	419
Lake	821	789	818
Slightly blurred images			
Lena	376	344	408
Sky	233	326	413
Lake	803	771	803

When blurred images are input to the metric system, the metric decrease as the intensity of the blurring is increased. The values decrease less as the sharpness of the image is continuously reduced. This is depicted in figure 8.4, which contain plots of the metric results from calculations with blurred images.

As shown in figure 8.5, when images are distorted with even low intensity noise, the median metric reaches a high value very fast. In the case of salt and pepper noise, the metric reaches the maximum already at very low density, while in the case of Gaussian noise, the metric grows less consistently towards the maximum value. Speckle noise does not seem to affect the metric to a large extent.

The tests with the artificial image in figure 5.2 shows that the metric reaches the highest absolute gradient value at the corners of the blocks, while it holds a steady value at the other block border areas. The metric is equal to zero at all other times. A test with a slightly smoothed version of the image shows a

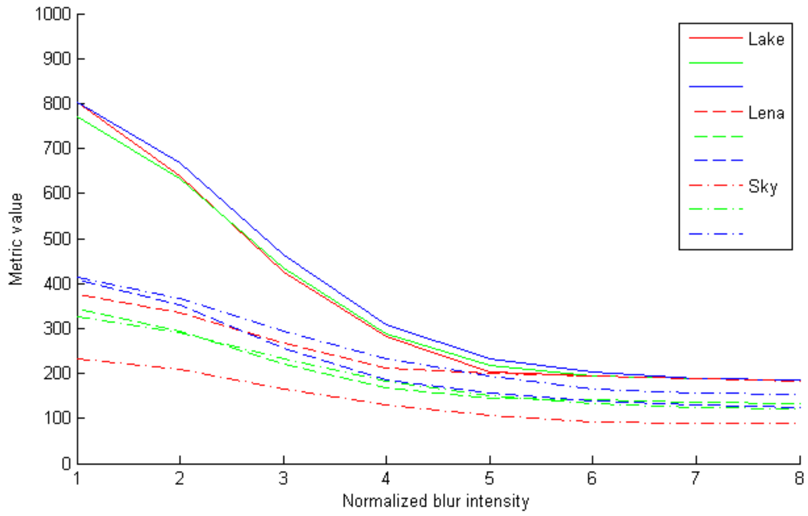


FIGURE 8.4: The plot shows the metric values as the intensity of blur is increased in all of the images in the test bench.

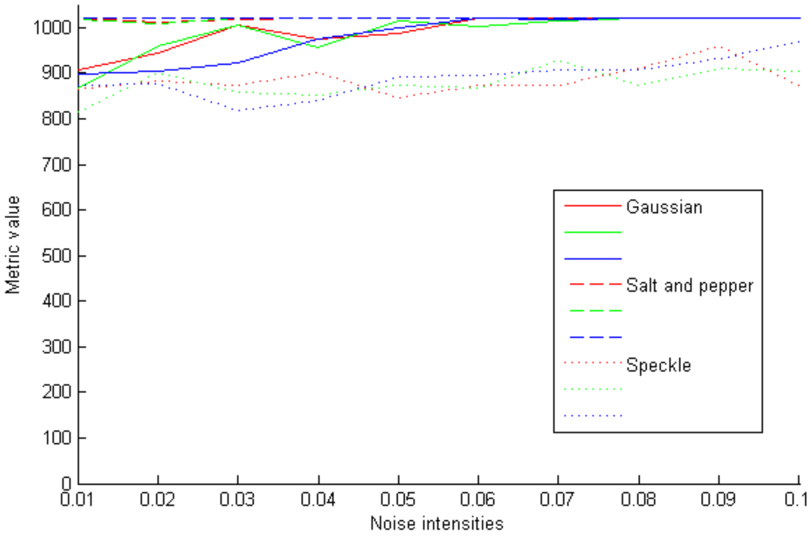


FIGURE 8.5: This figure depicts the metric output presented in table B.13. The x-axis is the variance of the noise added to the image, while the y-axis is the output from the metric.

reduction of the metric value at these changes significantly, making this a very valuable test when looking for blurred lines.

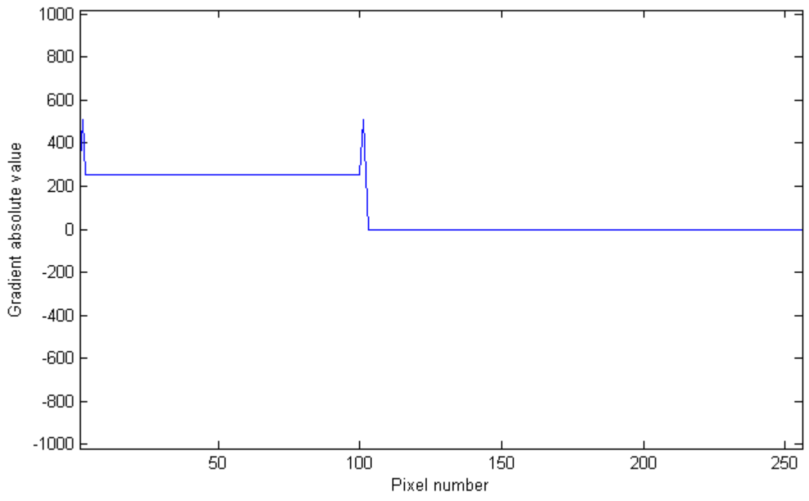


FIGURE 8.6: This plot shows the absolute values of the gradients from the image in figure 5.2. The corners of the blocks appear as large spikes in the plot. The large area containing values just above 200 is the border of a block. The areas containing only zero values contain no changes in pixel values.

8.1.5 Gaussian filtering metric

Table 8.3 shows the metric values calculated from the original test bench images. The metric is consistent between channels, but vary greatly between different images.

The metric system reacts to all types of random noise by increasing the metric value with increased distortion intensities. Blurred images result in decreased metric values.

TABLE 8.3: Results from the median metric when the original images (no introduced distortions) are used for testing.

Original images			
Channel	Red	Green	Blue
Lena	993 441	846 952	927 407
Sky	187 160	146 060	193 219
Lake	3 279 286	3 320 218	3 093 170
Slightly blurred images			
Lena	991 977	546 445	926 164
Sky	187 159	145 594	192 373
Lake	3 249 225	3 289 309	3 064 694

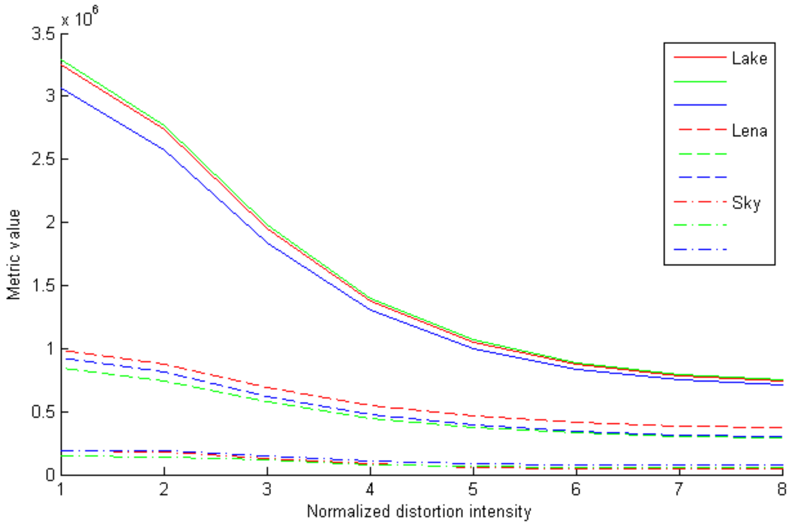


FIGURE 8.7: Plot showing the resulting metrics from the Gaussian assessment algorithm versus the blur intensity of the images using in the testing.

Table 8.4: Results from synthesizing some frequently used components in the metric systems in Quartus II.

Synthesis results for frequently used components			
	Row buffer	Averaging component	Histogram accumulator
Logical elements	32	573	6498
Registers	24	543	3072
Memory bits	30688	0	0
Embedded multipliers	0	0	0
Max frequency [MHz]	235	181	260

Table 8.5: Results from synthesizing the metric systems for the Altera Cyclone II FPGA in Quartus II.

Synthesis results for metric systems				
	A & D metric	Median metric	Gradient metric	Gaussian metric
Logical elements	1836	2042	927	653
Registers	1681	1885	727	543
Memory bits	57504	92064	92064	92064
Embedded multipliers	0	0	0	0
Max frequency [MHz]	162	176	183	203

8.2 Synthesis results

Table 8.5 and 8.4 shows the results from synthesising the systems for use in the Altera Cyclone II FPGA. The results show that the resource consumption of all metric systems are far below the maximum allowed use of any resource, as specified in chapter 5.1. In addition, the maximum frequency of all circuits are above the required pixel clock frequency used in the video resolution standard that has been specified.

Chapter 9

Discussion

The success of the VQA systems is examined in this chapter. The results from testing the algorithms in MATLAB, and the refinements these results point towards, are discussed in the first section. Some improvements regarding other algorithms for VQA and analyses of the metrics are also discussed. The next section debates the success of running the algorithms on an FPGA. The last section sums up the usefulness of the approach and evaluates the application area of the entire system.

9.1 Algorithm design and testing

Most of the algorithms developed for assessing the quality of video have been developed for use as no-reference assessment tools. However, in order to be able to test for special types of faults, such as poor vertical and horizontal sync and the video system's ability to maintain a linear or constant signal, some reduced-reference assessment tools have been proposed. The calculations of no-reference metrics for these cases, without any control on the input signal, is very complex since the behaviour we wish to study is unlikely to appear in any natural images. The reduced-reference tools provide valuable information regarding the mentioned artifacts, but does demand that the proper test pattern is input to the system and that the system is properly set up according to this pattern beforehand.

The no-reference video quality assessment tools will work with an unknown reference. Still, when combined with the artificial images presented, they will give even more information to the user. The reason for this is that the video assessment tools look for specific behaviour in the video. Unknown references contain unknown amounts of this behaviour, while the artificial images are designed with the exact behaviour needed for the metric systems to provide good information immediately. The test patterns allowing for the discovering of vertical and horizontal sync errors and the system's ability to uphold a linear signal are the only ones that are completely necessary in order to discover a fault. The other test patterns are included just to make the discovering of board-level failures easier and to make the metrics more reliant.

The testing of the quality assessment algorithms were done mainly using a test bench of four images contaminated with random noise and blur. Results from these tests show that there is a correlation between all of the metrics and the distortion they were designed for. In addition, all metrics correlate in some degree with other distortions. Most apparent is the gradient metric when utilized on images contaminated with salt and pepper noise. To be able to properly trace the metric results back to a board-level fault it is necessary to run the metrics together so that they can complement each other.

The results from the metric systems are consistent in that they always return a similar value when utilized on the same images with equal distortion intensities. On the other hand, the metrics vary greatly when utilized on different images. This can make the systems very hard to comprehend when utilized as pure no-reference systems. The metrics may also be hard to comprehend when they are not normalized or treated in order to contain the values in a smaller dynamic area. Provided you know the expected metric value, the metric can be used as is, but in a complete assessment system where the frames are unknown and not static, the raw metric data should be processed in a better way. Averaging, normalization and perhaps a merging of metrics are of the post processing that can be introduced.

The different metric values from the colour channels react in the same way for all of the tested images. It is also interesting to notice that results from analysing the sharpness in the three colour channels from the same images are very similar for all the natural images. In a refined system, this should be exploited by doing an analysis of the differences between the colour channels.

The main purpose of the video assessment system is to provide information on the state of the board-level components of the video device. The assessment tools have been developed specifically to output values that correlate with faults in the stream that are assumed to result from failures related to these board-level components. The assumptions have not been verified physically, but have been thoroughly investigated in theory. The success of the system relies on that the assumptions made are correct. The system should therefore be tested on a physical device, where the different failures are introduced, so that the assumptions can be properly verified.

Based on the test results of the metric systems, the algorithms will produce results that provides enough information for finding most board-level faults, at least when utilized together with the developed test patterns. However, issues related to the clock and other signal jitter are some of the faults that are not covered by the system, since their manifestations on the input signals are very hard to quantize correctly.

Further testing of the algorithms should be done with more specialized test patterns, such as a compilation of the test patterns presented. Utilizing such a test bench together with a fully developed system containing all of the proposed metrics will provide comprehensible results in a very fast way when doing testing at the production site and in the design phase of new systems. Compared to other metrics that are designed for running complex calculations on one processor at a time, this system is very well suited for the intended applications. However, compared to external test equipment, this metric can not achieve as good results. Therefore, if a comprehensive analysis of the perceptive quality of the video is needed, this system should not be utilized.

The Laplacian filter is known for being very sensitive to noise, and especially salt and pepper noise, since this noise generates very large gradients. If necessary, the filter can be combined with a Gaussian filter, which smooth out the image.

This leads to a more robust metric, but also causes the metric to lose some one of its advantages, since it can no longer determine if the system can operate with the largest frequencies.

There is also the possibility of replacing the Laplacian filter with a horizontal Sobel filter, a filter type that is commonly used for edge detection. There is no need for filtering in the vertical direction, since the loss of frequency components only happen as the lines of the frames are transmitted. The intensity of the edges could work as a replacement for, or along with the gradient intensity. The testing and implementation of this should be included as part of the future work.

9.2 VHDL modelling

The full implementation of a working metric system has not been modelled, as information on the control logic and similar aspects are not of importance to the task at hand. However, the verification has proven that the metric data from an image are exactly the same as the results from the modelling and testing of the algorithms.

By modelling and implementing the key metric components, we have proven that the algorithms will function within the specifications of the assignment. Verification of some relevant situations proves that the components work as they should, suggesting that a final implementation of a working system will function as intended. The real time capabilities of the system is also within specifications, providing a maximum clock frequency of at least 150 MHz. Since the target FPGA is made using relatively old technology, it is probable that newer FPGAs can provide even better real time capabilities with the same configuration. The memory consumption of the metrics are far below the requirements, even without any attempt of resource sharing.

The modelling of the key algorithm components in VHDL, and the simulation and synthesis of these models, proves that the algorithms will function as intended on an FPGA. It also proves that the metric systems are small enough for combining several metric calculations in one device, providing the collaboration desired in order to find component faults. The simplicity of the algorithms and the possibility for reusing the components in similar metric systems gives the indication that a system consisting of the proposed metrics not only will fit and work in an FPGA, but also leave room for other systems or an expansion in the number of metric systems.

There has been made no attempt on sharing resources and components between the metric systems. This is, however, very possible, since the different metrics are based on very much the same principles. For example, there is only need for one row buffer for the metric systems utilizing 3×3 spatial filters.

Some further implementation is required in order to make a functional quality assessment system. The control logic must be designed, allowing for proper reception of new frames and proper calculation of correct data values. In addition, the entire system should be simulated, including metric system components for each color channel. The models must also be tested on an FPGA, verifying that the synthesis is correct and according to the specifications.

Since all VHDL modules are written with the frame size and data channel size as generic parameters, it is possible to use the metric systems with any resolution. The systems are however only tested with parameters equal to that of the tests signals. There is therefore a necessity of verifying that the systems work with all common resolutions and that the systems still are within the required specifications.

9.3 Final discussion

Using analyses of video specifically in order to detect failures in board-level components is something which has not been done before. This report therefore presents a new perspective on the field of video metric measurement. Although the approach is unique, there are several algorithms that can detect some of the relevant distortions and artifacts. However, the systems presented here are also designed specifically towards FPGAs. This provides some opportunities that are not possible on other devices. First, several different metrics can be run simultaneously in a single device. This allows for the calculation of a large number of simple metrics, providing great coverage of the artifacts, and thereby increasing the possibility of tracing an artifact back to the correct hardware fault. Second, the FPGA is often the first device in the video processing path, meaning the signal that is fed into the video assessment system is uncontaminated by any processing. This provides shorter setup time for the test that are to be performed, since the FPGA is the only device that must be specifically configured.

Table 9.1 shows how the different assessment tools and metric systems cover the mentioned board-level component faults. In some cases there are several possible causes of a specific distortion. In order to distinguish between these cases, the distortions must be modelled better, for instance by testing an actual video system with the specific faults. There are also some distortions that are covered by the same metrics. In these cases, the appropriate tests consist of a combination of metrics. Salt and pepper noise, Gaussian noise and speckle noise are examples of distortions that require combined metrics, since the Median metric and the A & D metric responds to all cases, while the Gradient metric response can be distinguished between the three cases. In addition, the fact that the A & D metric does not respond to blurring of frames can be used to ensure that an image is distorted only by blur.

Table 9.1: Overview of board-level failures, their associated distortions and the metrics that are able to distinguish them.

Metric system distortion coverage		
Hardware fault	**Associated failures**	**Appropriate tests**
Faulty filter components	Low resistance to noise	Median metric and A & D metric.
	Inadequate bandwidth	Gradient metric and the Gaussian filtering metric in combination with the A & D metric.
	Bad frequency response	Histogram analysis in combination with test patterns.
Clock jitter	Blurring	Gradient metric and Gaussian filtering metric.
	Gaussian noise	A & D metric and Median metric.
Sync signal jitter	Frame displacement	Specialized reduced-reference assessment tools.
Voltage failures	Difficulties in reaching specific values	Histogram analysis and data value analysis.
	Sporadic IC failure	Histogram analysis.
Electromagnetic noise	Gaussian noise	A & D metric and Median metric.
	Salt and pepper noise	A & D metric, Median metric and Gradient metric.
Camera noise	Speckle noise	A & D metric and median metric combined with Gradient metric.
	Salt and pepper noise	A & D metric and median metric combined with Gradient metric.
Stuck-at faults	Loss of specific values	Specialized assessment tool.

Chapter 10

Concluding remarks

The video metric measurements presented in this report constitutes new ways of discovering failures in the hardware of analogue video systems. By using an FPGA as a processing device and by focusing on utilizing several simple metrics in parallel, hardware failures that otherwise requires experienced engineers or expensive equipment to be discovered are shown to be quantifiable by a single system operating in real time.

Although a fully functional video metric system has not been developed, the results from metric testing and component analyses show that the proposed system is fully usable and possible to implement on even small, low-range FPGAs. The system will function with all video systems utilizing an FPGA and transmission of video through analogue channels, making it reusable with most advanced analogue video system platforms.

Most of the metric systems are usable without knowledge of the input signal, although the resulting metric values are much more comprehensible when the output or the input is known beforehand. Assessment tools developed for reduced-reference testing are provided to be able to test for hardware faults that cannot be discovered in any other fashion.

10.1 Future work

The future work necessary for making a functional VQA system include the following points:

- Full implementation of the metric systems and the top level VQA system.
- Behavioural verification on an FPGA.
- Test with actual physical board-level failures.

It is also advised to extend the VQA system with the following metric systems:

- Assessment based on bilateral filters or Frost-filters for quantization of speckle noise.
- Analyses based on the correlations of neighbouring bins in a histogram.
- Analyses based on the differences in metric values from different colour channels.

Appendix A

MATLAB scripts

A.1 Modelling of median filter metric

./Scripts/noise_estimation_medfilt.m

```
close all;
clear all;

%% Estimation of random noise in a video sequence using the median
    filter-based metric.

% Specify which image to be tested.
image_name = 'sky';
noise_type = 'blur';

% Instantiate the metric result vectors.
metric_red = zeros(10,1);
metric_green = zeros(10,1);
metric_blue = zeros(10,1);
psnr_value_red = zeros(10,1);

% Load image
original_image = imread([image_name '.bmp']);
[rows, cols, dims] = size(original_image);

% Metric index instantiation.
index = 1;
```

```
% Loop for testing with different noise intensities.
for noise_intensity = 0.1:0.1:1

    disp = ['Calculating median metric for noise intensity ' num2str
    (noise_intensity) '.'];
    display(disp);

    % Read in generated frame with the specified distortions.
    image_string = [image_name '_' noise_type '
    _noise_with_intensity_' num2str(noise_intensity) '_firstFrame.
    bmp'];

    noisy_image = imread(image_string);

    [rows, cols, dims] = size(noisy_image);

    % Separate the channels:
    red = noisy_image(:,:,1);
    green = noisy_image(:,:,2);
    blue = noisy_image(:,:,3);

    % Filter each channel through a 2D median filter:
    filtered_red = medfilt2(red);
    filtered_green = medfilt2(green);
    filtered_blue = medfilt2(blue);

    % Find the absolute differences:
    diff_red = abs(double(red)-double(filtered_red));
    diff_green = abs(double(green)-double(filtered_green));
    diff_blue = abs(double(blue)-double(filtered_blue));

    % Find the sum of absolute differences:
    for i = 1:rows
        for j = 1:cols
            metric_red(index) = metric_red(index) + int32(diff_red(i
    ,j));
            metric_green(index) = metric_green(index) + int32(
    diff_green(i,j));
            metric_blue(index) = metric_blue(index) + int32(
    diff_blue(i,j));
        end
    end

    % Calculate the PSNR for comparison.
    psnr_value_red(index) = PSNR(original_image(:,:,1),noisy_image
    (:,:,1));

    % Update metric vector index.
    index = index + 1;

end

% Save metric data to a text file as a latex table.

    % Noise variance.
    nv = 0.1:0.1:1;
    nv = nv';

    % Metric data as a matrix.
    data = [nv metric_red metric_green metric_blue];

    % Call to function for saving data to latex format in file.
    saveDataToLatex(data, 'noise_estimation_001_01', 'txt');
```

A.2 Modelling of A & D metric

./Scripts/noise_estimation_ad.m

```
close all;
clear all;

%% Estimation of random noise in a video sequence (tests over two
    frames) using A&D metric.

% Specify which image to be tested.
image_name = 'lena';
noise_type = 'blur';

% Instantiate the metric result vectors.
% 10 tests are performed.
metric_red = zeros(10,1);
metric_green = zeros(10,1);
metric_blue = zeros(10,1);
psnr_value_red_first = zeros(10,1);
psnr_value_red_second = zeros(10,1);

% Load image.
original_image = imread([image_name '.bmp']);

% Metric index instantiation.
index = 1;

% Loop for testing with different noise intensities.
for noise_intensity = 0.1:0.1:0.1

    disp = ['Calculating metric for noise variance ' num2str(
    noise_intensity) '.'];
    display(disp);

    % Read in two generated frames with the specified distortions.

    image_string_firstFrame = [image_name '_' noise_type '
    _noise_with_intensity_' num2str(noise_intensity) '_firstFrame.
    bmp'];
    image_string_secondFrame = [image_name '_' noise_type '
    _noise_with_intensity_' num2str(noise_intensity) '_secondFrame.
    bmp'];

    noisy_first = imread(image_string_firstFrame);
    noisy_second = imread(image_string_firstFrame);

    [rows, cols, dims] = size(noisy_first);

    % Metric calculations.
    % Instantiate the previous row value registers.
    red_row_registers = zeros(rows,1);
    green_row_registers = zeros(rows,1);
    blue_row_registers = zeros(rows,1);

    % Instantiate the current row accumulation registers.
    red_row_acc_registers = int32(0);
    green_row_acc_registers = int32(0);
    blue_row_acc_registers = int32(0);

    % Loop for accessing each pixel for first frame calculations.
    for i = 1:rows
        for j = 1:cols
            % Accumulate the values in the current row.
            red_row_acc_registers = red_row_acc_registers + int32(
    noisy_first(i,j,1));
            green_row_acc_registers = green_row_acc_registers +
    int32(noisy_first(i,j,2));
            blue_row_acc_registers = blue_row_acc_registers + int32(
    noisy_first(i,j,3));
        end

        % Store the accumulated values in the previous row value
    register.
        red_row_registers(i) = red_row_acc_registers;
        green_row_registers(i) = green_row_acc_registers;
        blue_row_registers(i) = blue_row_acc_registers;

        % Reset the registers.
        red_row_acc_registers = 0;
        green_row_acc_registers = 0;
        blue_row_acc_registers = 0;
    end

    % Loop for accessing each pixel for second frame calculations.
    for k = 1:rows
        for l = 1:cols
            % Accumulate the values in the current row.
            red_row_acc_registers = red_row_acc_registers + int32(
    noisy_second(k,l,1));
            green_row_acc_registers = green_row_acc_registers +
    int32(noisy_second(k,l,2));
            blue_row_acc_registers = blue_row_acc_registers + int32(
    noisy_second(k,l,3));
        end

        % Store the absolute difference between the accumulated
    values of the first
        % frame and the second frame.
        metric_red(index) = metric_red(index) + abs(
    red_row_acc_registers - red_row_registers(k));
        metric_green(index) = metric_green(index) + abs(
    green_row_acc_registers - green_row_registers(k));
        metric_blue(index) = metric_blue(index) + abs(
    blue_row_acc_registers - blue_row_registers(k));

        % Reset the registers.
        red_row_acc_registers = 0;
        green_row_acc_registers = 0;
        blue_row_acc_registers = 0;
    end

    % Calculate the PSNR for comparrison.
    psnr_value_red_first(index) = PSNR(original_image(:,:,1),
    noisy_first(:,:,1));
    psnr_value_red_second(index) = PSNR(original_image(:,:,1),
    noisy_second(:,:,1));

    % Update the metric vector index.
    index = index + 1;
end

% Save metric data to a text file as a latex table.

    % Noise variance.
    nv = 1:10;
    nv = nv';

    % Metric data as a matrix.
    data = [nv metric_red metric_green metric_blue];

    % Call to function for saving data to latex format in file.
    saveDataToLatex(data, 'noise_estimation_001_01', 'txt');
```

A.3 Modelling of gradient metric

./Scripts/blur_estimation_laplace.m

```
close all;
clear all;

%% Estimation of blur in a image with increasing loss of sharpness.

% Specify which image to be tested.
image_name = 'blocks';
noise_type = 'blur';

% Instantiate the metric result vectors.
metric_red = zeros(10,1);
metric_green = zeros(10,1);
metric_blue = zeros(10,1);
psnr_value_red = zeros(10,1);

% Load image.
original_image = imread([image_name '.bmp']);

% Metric index instantiation.
index = 1;

% Loop for testing with different distortion intensities.
for noise_intensity = 0.1:0.1:1

    disp = ['Calculating blur metric for distortion intensity '
    num2str(noise_intensity) '.'];
    display(disp);

    % Read in a frame generated with the specified distortions.
    image_string_frame = 'blocks.bmp'%[image_name '_' noise_type '
    _noise_with_intensity_' num2str(noise_intensity) '_firstFrame.
    bmp'];
    frame = imread(image_string_frame);
    [rows, cols, dims] = size(frame);

    % Find the level of blurriness/sharpness by using a Laplace
    filter.

    % Laplace operator:
    template = [0 1 0; 1 -4 1; 0 1 0];
    % Filter the image through the Laplace filter
    filtered_frame_red = conv2(double(frame(:,:,1)), double(template));
    filtered_frame_green = conv2(double(frame(:,:,2)), double(template));
    filtered_frame_blue = conv2(double(frame(:,:,3)), double(template));

    % Update the metric result vector.
    metric_red(index) = max(max(abs(filtered_frame_red)));
    metric_green(index) = max(max(abs(filtered_frame_green)));
    metric_blue(index) = max(max(abs(filtered_frame_blue)));

    % Calculate the PSNR for comparison.
    psnr_value_red(index) = PSNR(original_image(:,:,1),frame(:,:,1));

    % Update the metric index.
    index = index + 1;

end

% Save metric data to a text file as a latex table.

    % Noise variance.
    nv = 0.01:0.01:0.1;
    nv = nv';

    % Metric data as a matrix.
    data = [nv metric_red metric_green metric_blue];

    % Call to function for saving data to latex format in file.
    saveDataToLatex(data, 'data_to_latex', 'txt');
```

A.4 Modelling of smoothing metric

./Scripts/blur_estimation_gaussian.m

```
close all;
clear all;

%% Estimation of blurring in a a video frame.

% Specify which image to be tested.
image_name = 'lena';
noise_type = 'gaussian';

% Instantiate the metric result vectors.
% 10 tests are performed.
metric_red = zeros(10,1);
metric_green = zeros(10,1);
metric_blue = zeros(10,1);
psnr_value_red_first = zeros(10,1);

% Load image.
original_image = imread([image_name '.bmp']);

% Metric index instantiation.
index = 1;

% Loop for testing with different noise intensities.
for noise_intensity = 0.01:0.01:0.1

    disp = ['Calculating metric for noise variance ' num2str(
    noise_intensity) '.'];
    display(disp);

   % Read in two generated frames with the specified distortions.

    image_string_frame = 'lake.bmp'%[image_name '_' noise_type '
    _noise_with_variance_' num2str(noise_intensity) '_firstFrame.bmp
    '];

    frame = imread(image_string_frame);

    [rows, cols, dims] = size(frame);

   % Separate the channels:
```

```
red = frame(:,:,1);
green = frame(:,:,2);
blue = frame(:,:,3);

% Metric calculations.
% Gaussian operator:
template = [1/16 2/16 1/16; 2/16 4/16 2/16; 1/16 2/16 1/16];

% Filter the image through the Gaussian filter.
filtered_frame_red = conv2(double(red), double(template));
filtered_frame_green = conv2(double(green), double(template));
filtered_frame_blue = conv2(double(blue), double(template));

filtered_frame_red = uint8(filtered_frame_red(2:rows+1, 2:cols
+1));
filtered_frame_green = uint8(filtered_frame_green(2:rows+1, 2:
cols+1));
filtered_frame_blue = uint8(filtered_frame_blue(2:rows+1, 2:cols
+1));

% Find the differences:
diff_red = abs(double(frame(:,:,1)) - double(filtered_frame_red)
);
diff_green = abs(double(frame(:,:,2)) - double(
filtered_frame_green));
diff_blue = abs(double(frame(:,:,3)) - double(
filtered_frame_blue));

% Find the SAD.
for i = 1:rows
    for j = 1:cols
        metric_red(index) = metric_red(index) + int32(diff_red(i
,j));
        metric_green(index) = metric_green(index) + int32(
diff_green(i,j));
        metric_blue(index) = metric_blue(index) + int32(
diff_blue(i,j));
    end
end

% Calculate the PSNR for comparrison.
psnr_value_red_first(index) = PSNR(original_image(:,:,1),frame
(:,:,1));
```

```
    % Update the metric vector index.
    index = index + 1;
end

% Save metric data to a text file as a latex table.

    % Noise variance.
    nv = 0.01:0.01:0.1;
    nv = nv';

    % Metric data as a matrix.
    data = [nv metric_red metric_green metric_blue];

    % Call to function for saving data to latex format in file.
    saveDataToLatex(data, 'noise_estimation_001_01', 'txt');
```

Appendix B

VQA systems test results

B.1 Results from PSNR analyses

Table B.1: Results from PSRN using natural images with different levels of Gaussian noise introduced.

PSNR			
Gaussian noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	20.17	20.20	20.28
0.02	17.33	17.43	17.55
0.03	15.76	15.86	16.02
0.04	14.67	14.76	14.96
0.05	13.88	13.97	14.18
0.06	13.22	13.35	13.52
0.07	12.70	12.79	12.97
0.08	12.27	12.36	12.53
0.09	11.86	11.97	12.14
0.10	11.56	11.62	11.79
Sky			
0.01	20.04	20.02	20.11
0.02	17.11	17.04	17.36
0.03	15.46	15.32	15.85
0.04	14.34	14.14	14.84
0.05	13.54	13.29	14.07
0.06	12.91	12.63	13.43
0.07	12.37	12.08	12.90
0.08	11.96	11.64	12.46
0.09	11.62	11.30	12.06
0.10	11.26	10.99	11.74
Lake			
0.01	20.47	20.61	21.19
0.02	17.79	17.85	18.43
0.03	16.22	16.31	16.82
0.04	15.17	15.24	15.73
0.05	14.35	14.41	14.87
0.06	13.69	13.74	14.17
0.07	13.14	13.21	13.60
0.08	12.67	12.71	13.09
0.09	12.27	12.35	12.70
0.10	11.93	11.96	12.31

Table B.2: Results from PSRN using natural images with different levels of salt & pepper noise introduced.

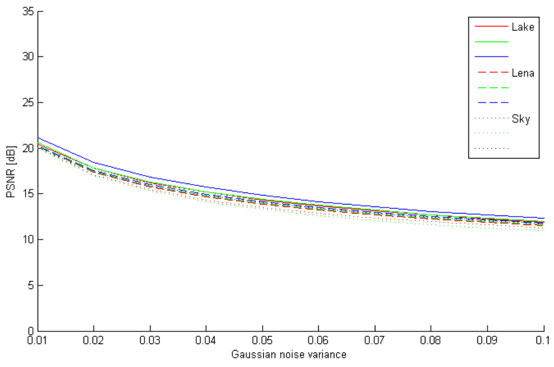
PSNR			
Salt & pepper noise			
Noise density	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	25.04	25.04	25.11
0.02	22.32	22.08	21.87
0.03	20.39	20.39	20.19
0.04	19.27	19.13	18.84
0.05	18.21	18.09	18.00
0.06	17.44	17.28	17.25
0.07	16.76	16.65	16.46
0.08	16.23	16.07	15.93
0.09	15.67	15.53	15.39
0.10	15.16	15.16	14.98
Sky			
0.01	25.68	25.94	24.87
0.02	22.60	22.98	22.06
0.03	20.94	21.20	20.33
0.04	19.60	20.02	19.05
0.05	18.67	19.08	18.12
0.06	17.77	18.28	17.37
0.07	17.23	17.55	16.59
0.08	16.66	16.96	16.05
0.09	16.08	16.40	15.49
0.10	15.65	15.99	15.10
Lake			
0.01	24.69	24.68	24.17
0.02	21.68	21.54	21.12
0.03	19.99	19.91	19.35
0.04	18.67	18.70	18.09
0.05	17.72	17.72	17.11
0.06	16.97	16.91	16.29
0.07	16.30	16.24	15.64
0.08	15.68	15.56	15.08
0.09	15.20	15.10	14.64
0.10	14.69	14.65	14.10

Table B.3: Results from PSRN using natural images with different levels of speckle noise introduced.

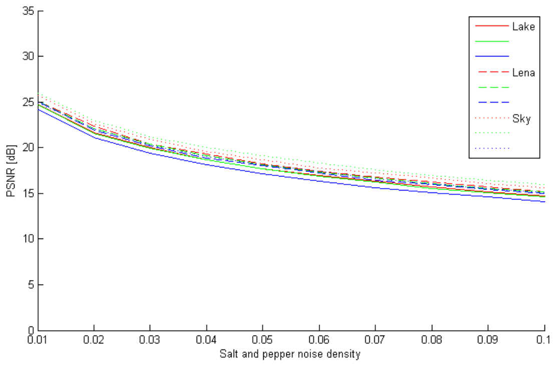
PSNR			
Speckle noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	24.98	29.00	30.51
0.02	22.08	25.99	27.50
0.03	20.35	24.22	25.73
0.04	19.17	22.99	24.49
0.05	18.28	22.03	23.52
0.06	17.54	21.22	22.74
0.07	16.92	20.58	22.08
0.08	16.40	20.00	21.49
0.09	15.93	19.50	20.99
0.10	15.53	19.03	20.52
Sky			
0.01	29.15	25.69	22.56
0.02	26.14	22.70	19.57
0.03	24.38	20.93	17.84
0.04	23.14	19.67	16.69
0.05	22.16	18.72	15.83
0.06	21.39	17.91	15.16
0.07	20.70	17.25	14.60
0.08	20.12	16.66	14.14
0.09	19.61	16.16	13.74
0.10	19.15	15.69	13.39
Lake			
0.01	25.41	25.29	25.76
0.02	22.67	22.60	22.84
0.03	21.05	20.99	21.17
0.04	19.92	19.86	19.94
0.05	19.06	18.97	19.03
0.06	18.34	18.25	18.30
0.07	17.73	17.63	17.66
0.08	17.24	17.09	17.09
0.09	16.78	16.65	16.59
0.10	16.38	16.21	16.18

Table B.4: FEIL Results from PSNR calculations using natural images with different levels of blurring.

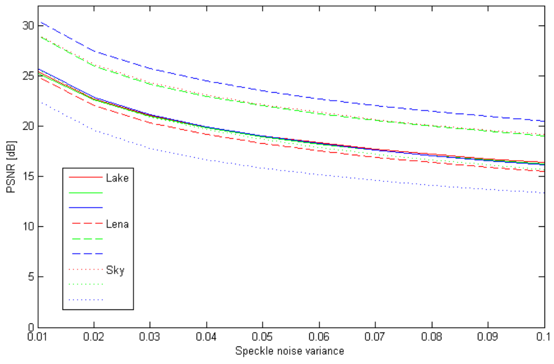
PSNR			
Blurring			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.3	68.01	73.07	70.42
0.4	46.24	48.06	47.04
0.5	38.30	40.23	39.26
0.6	34.74	36.74	35.86
0.7	32.95	34.99	34.17
0.8	31.96	34.02	33.24
0.9	31.34	33.42	32.68
10	30.93	33.03	32.31
Sky			
0.3	93.37	71.52	68.52
0.4	56.38	53.74	50.80
0.5	47.33	45.09	42.18
0.6	43.72	41.21	38.33
0.7	41.82	39.23	36.35
0.8	40.75	38.11	35.22
0.9	40.09	37.40	34.51
1.0	39.66	36.94	34.05
Lake			
0.3	58.28	58.15	58.56
0.4	38.56	38.44	38.77
0.5	30.79	30.66	30.99
0.6	27.52	27.37	27.70
0.7	25.96	25.80	26.13
0.8	25.11	24.95	25.27
0.9	24.61	24.44	24.76
1.0	24.29	24.11	24.43



(A) Plot of the PSNR values from table B.1.



(B) Plot of the PSNR values from table B.2.



(C) Plot of the PSNR values from table B.3.

FIGURE B.1: The three plots show how the PSNR evolves as the intensity of Gaussian noise, salt & pepper noise and speckle noise increases.

B.2 Results from testing the A & D metric

Table B.5: Results from simulating the *accumulate and differentiate* metric using natural images with different levels of Gaussian noise introduced.

Accumulate and differentiate			
Gaussian noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	334684	330662	331414
0.02	466105	442788	425789
0.03	561571	504404	524452
0.04	625327	626501	607679
0.05	699278	622373	647622
0.06	707652	717766	696144
0.07	768655	736291	721965
0.08	838667	760053	785522
0.09	793399	830853	757306
0.10	918612	838086	852211
Sky			
0.01	242668	249930	247850
0.02	333377	342969	344006
0.03	400671	425026	390437
0.04	464236	490683	425486
0.05	512504	520438	468611
0.06	564553	537663	520748
0.07	561937	586286	544413
0.08	677665	590814	529439
0.09	654674	669207	583883
0.10	653771	662077	574665
Lake			
0.01	390186	376567	326101
0.02	503126	496457	459724
0.03	609025	630384	566524
0.04	704280	694316	644091
0.05	765685	713730	693002
0.06	758743	835483	758533
0.07	893655	853045	779963
0.08	930516	913447	864386
0.09	913433	955711	911517
0.10	964555	1010820	961209

TABLE B.6: Results from simulating the *accumulate and differentiate* metric using natural images with different levels of salt & pepper noise.

Accumulate and differentiate			
Salt & pepper noise			
Noise density	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	191551	178918	188649
0.02	248940	260724	273242
0.03	324408	337767	344010
0.04	349694	370443	361123
0.05	397113	391388	417737
0.06	423619	441191	452206
0.07	450773	516700	482878
0.08	505202	489954	543323
0.09	560966	547529	560895
0.10	596789	591578	580511
Sky			
0.01	132655	116235	133226
0.02	187116	178913	197764
0.03	222323	216069	225012
0.04	239588	242399	263938
0.05	298546	288007	296991
0.06	321228	283584	346003
0.07	322815	322609	367539
0.08	351343	341617	371554
0.09	365928	345714	398157
0.10	397170	369705	395519
Lake			
0.01	226861	225958	241283
0.02	328354	328300	334554
0.03	388571	418121	446960
0.04	460313	449038	506641
0.05	510779	501314	562893
0.06	577045	612069	589616
0.07	638158	653929	642952
0.08	652647	635766	668095
0.09	699095	714336	759944
0.10	758484	723237	770447

Table B.7: Results from simulating the *accumulate and differentiate* metric using natural images with different levels of speckle noise.

Accumulate and differentiate			
Speckle noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	184416	124087	93990
0.02	254774	168264	133622
0.03	325261	218504	171667
0.04	371029	235214	196788
0.05	411125	264996	208636
0.06	456683	308200	227730
0.07	481731	306382	255053
0.08	481009	338283	282435
0.09	532456	349358	296499
0.10	555515	383830	313606
Sky			
0.01	78936	132117	163696
0.02	102544	175991	250053
0.03	148373	221192	317371
0.04	167275	254204	365534
0.05	192712	301225	392650
0.06	214531	315783	411659
0.07	225722	350805	446511
0.08	235643	361931	460386
0.09	261010	378169	531422
0.10	268234	396573	534966
Lake			
0.01	196903	191364	177304
0.02	276574	288752	238592
0.03	323568	338643	286589
0.04	373821	373717	330994
0.05	399699	395051	358645
0.06	427678	455010	377425
0.07	500371	477257	440884
0.08	493617	502474	458405
0.09	536811	537894	495496
0.10	576543	540459	537711

B.3 Results from testing the median filter metric

Table B.8: Results from simulating the *median filter-based* metric using natural images with different levels of Gaussian noise introduced.

Median filter and SAD			
Gaussian noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	4913313	4890370	4877412
0.02	6762193	6663023	6626283
0.03	8061861	7963160	7854952
0.04	9131804	8993836	8830205
0.05	9986123	9849536	9622042
0.06	10749044	10544812	10326682
0.07	11445561	11240398	10970763
0.08	12005326	11814176	11512375
0.09	12605568	12333161	12027002
0.10	13026455	12815593	12475062
Sky			
0.01	3690953	3690724	3676566
0.02	5159099	5197207	5081958
0.03	6291165	6378118	6060828
0.04	7202224	7324333	6839968
0.05	7902416	8098197	7453557
0.06	8566118	8798671	7991072
0.07	9111458	9437555	8496039
0.08	9602881	9931748	8913510
0.09	9972337	10404944	9317002
0.10	10450917	10828605	9641268
Lake			
0.01	7832916	7744973	6787682
0.02	9896935	9781194	8564327
0.03	11441746	11272618	9894879
0.04	12645097	12444621	11000850
0.05	13738173	13512697	11927856
0.06	14644436	14448499	12796746
0.07	15437628	15185811	13517000
0.08	16224109	15971446	14224020
0.09	16891162	16616612	14804162
0.10	17442700	17193317	15421615

TABLE B.9: Results from simulating the *median filter-based* metric using natural images with different levels of salt & pepper noise.

Median filter and SAD			
Salt & pepper noise			
Noise density	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	1253892	1159403	1245412
0.02	1555204	1480634	1586747
0.03	1910087	1794591	1904827
0.04	2216150	2129376	2252859
0.05	2565114	2478555	2542296
0.06	2878066	2809993	2844384
0.07	3215302	3128173	3225652
0.08	3517618	3455104	3528894
0.09	3865278	3794746	3867057
0.10	4227065	4079874	4154346
Sky			
0.01	417799	357488	387752
0.02	673480	608432	628556
0.03	915817	866291	880934
0.04	1182300	1102258	1139513
0.05	1418894	1344665	1380311
0.06	1700985	1597547	1620974
0.07	1919299	1867381	1913520
0.08	2163944	2131316	2150473
0.09	2432307	2409866	2428063
0.10	2673469	2632558	2646827
Lake			
0.01	3907658	3952600	3665144
0.02	4349528	4408883	4104084
0.03	4760735	4808378	4543031
0.04	5218099	5235151	4979705
0.05	5657254	5664657	5421495
0.06	6064378	6102262	5867417
0.07	6507318	6535193	6312280
0.08	6969966	7027328	6722281
0.09	7390852	7434459	7109451
0.10	7852429	7851400	7602769

Table B.10: Results from simulating the *median filter-based* metric using natural images with different levels of speckle noise.

Median filter and SAD			
Speckle noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	2915053	1892599	1719823
0.02	3924058	2507192	2202158
0.03	4732492	3001224	2613471
0.04	5400977	3419331	2945763
0.05	5968986	3798061	3260968
0.06	6476285	4141681	3543010
0.07	6931635	4452297	3804218
0.08	7341124	4728217	4033519
0.09	7759832	5015447	4258372
0.10	8085022	5281904	4489150
Sky			
0.01	1366819	2036942	2919382
0.02	1932455	2865980	4134893
0.03	2349996	3506148	5037761
0.04	2726061	4062721	5749638
0.05	3039261	4524007	6373510
0.06	3311711	4981862	6872448
0.07	3606207	5376821	7311647
0.08	3838158	5748848	7746472
0.09	4064583	6078712	8099165
0.10	4290988	6413692	8413304
Lake			
0.01	5487479	5587482	4861616
0.02	6473436	6511418	5679942
0.03	7224899	7256319	6329385
0.04	7864370	7914399	6918206
0.05	8419499	8451622	7392018
0.06	8917608	8973146	7864382
0.07	9396185	9421179	8274878
0.08	9813727	9894801	8690871
0.09	10219168	10285924	9100667
0.10	10601818	10686036	9393059

TABLE B.11: Results from simulating the *median filter-based* metric using natural images with different levels of blurring.

Median filter and SAD			
Blurring			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.30	911084	822685	927834
0.40	781209	706382	794164
0.50	568956	509720	573258
0.60	416170	369014	415405
0.70	325514	284219	320803
0.80	276798	239439	270769
0.90	250611	215149	243540
1.00	236076	201860	228569
Sky			
0.30	167821	101556	124273
0.40	163679	101065	123526
0.50	105597	80736	88689
0.60	67139	48132	54114
0.70	44494	30578	35059
0.80	33870	22052	26204
0.90	28798	16804	21178
1.00	27192	16489	20497
Lake			
0.30	3438949	3488204	3202801
0.40	2855667	2902961	2651254
0.50	1963643	2002786	1821165
0.60	1318460	1351312	1218971
0.70	955009	981778	882770
0.80	770420	792359	712347
0.90	679151	699244	628489
1.00	637483	655617	589173

B.4 Results from testing the gradient metric

TABLE B.12: Results from simulating the *Laplace filter-based* metric using natural images with different levels of Gaussian noise.

Absolute value of gradient			
Gaussian noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	598.00	537.00	649.00
0.02	708.00	724.00	744.00
0.03	900.00	848.00	819.00
0.04	961.00	959.00	937.00
0.05	925.00	978.00	976.00
0.06	1011.00	988.00	993.00
0.07	1011.00	1000.00	1020.00
0.08	1011.00	1019.00	1020.00
0.09	1013.00	1020.00	1020.00
0.10	1020.00	1020.00	1020.00
Sky			
0.01	543.00	565.00	501.00
0.02	760.00	684.00	689.00
0.03	856.00	781.00	806.00
0.04	945.00	947.00	895.00
0.05	953.00	953.00	960.00
0.06	1014.00	999.00	984.00
0.07	1008.00	958.00	1020.00
0.08	1020.00	998.00	1020.00
0.09	1020.00	994.00	1020.00
0.10	1020.00	984.00	1020.00
Lake			
0.01	906.00	865.00	897.00
0.02	944.00	960.00	904.00
0.03	1007.00	1007.00	922.00
0.04	975.00	956.00	975.00
0.05	986.00	1015.00	1000.00
0.06	1020.00	1003.00	1020.00
0.07	1020.00	1016.00	1019.00
0.08	1020.00	1020.00	1020.00
0.09	1020.00	1020.00	1020.00
0.10	1020.00	1020.00	1020.00

TABLE B.13: Results from simulating the *Laplace filter-based* metric using natural images with different levels of salt & pepper noise.

Absolute value of gradient			
Salt & pepper noise			
Noise density	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	1001.00	930.00	936.00
0.02	1011.00	953.00	931.00
0.03	1013.00	958.00	957.00
0.04	1006.00	952.00	969.00
0.05	1011.00	968.00	960.00
0.06	1012.00	974.00	958.00
0.07	1018.00	980.00	968.00
0.08	1016.00	962.00	996.00
0.09	1020.00	982.00	975.00
0.10	1020.00	970.00	984.00
Sky			
0.01	816.00	724.00	876.00
0.02	813.00	729.00	888.00
0.03	877.00	816.00	914.00
0.04	864.00	810.00	928.00
0.05	926.00	816.00	921.00
0.06	878.00	896.00	970.00
0.07	944.00	909.00	926.00
0.08	943.00	876.00	965.00
0.09	934.00	909.00	963.00
0.10	946.00	900.00	1020.00
Lake			
0.01	1020.00	1019.00	1020.00
0.02	1011.00	1009.00	1020.00
0.03	1019.00	1020.00	1020.00
0.04	1020.00	1020.00	1020.00
0.05	1020.00	1020.00	1020.00
0.06	1020.00	1020.00	1020.00
0.07	1020.00	1020.00	1020.00
0.08	1020.00	1020.00	1020.00
0.09	1020.00	1020.00	1020.00
0.10	1020.00	1020.00	1020.00

Table B.14: Results from simulating the *Laplace filter-based* metric using natural images with different levels of speckle noise.

Absolute value of gradient			
Speckle noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	494.00	388.00	464.00
0.02	540.00	469.00	486.00
0.03	552.00	494.00	597.00
0.04	573.00	601.00	531.00
0.05	606.00	517.00	638.00
0.06	657.00	653.00	599.00
0.07	646.00	597.00	724.00
0.08	686.00	663.00	689.00
0.09	704.00	676.00	630.00
0.10	699.00	668.00	739.00
Sky			
0.01	273.00	321.00	536.00
0.02	278.00	395.00	581.00
0.03	312.00	433.00	557.00
0.04	364.00	469.00	573.00
0.05	385.00	494.00	658.00
0.06	450.00	597.00	649.00
0.07	420.00	633.00	653.00
0.08	424.00	634.00	689.00
0.09	495.00	672.00	795.00
0.10	480.00	734.00	742.00
Lake			
0.01	864.00	814.00	873.00
0.02	882.00	901.00	877.00
0.03	872.00	859.00	817.00
0.04	902.00	852.00	839.00
0.05	847.00	873.00	893.00
0.06	873.00	868.00	895.00
0.07	873.00	930.00	906.00
0.08	911.00	875.00	908.00
0.09	960.00	910.00	931.00
0.10	869.00	905.00	970.00

Table B.15: Results from simulating the *Laplace filter-based* metric using natural images with different levels of blurring.

Absolute value of gradient			
Blurring			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.03	376.00	344.00	408.00
0.04	334.00	295.00	351.00
0.05	269.00	221.00	257.00
0.06	213.00	169.00	186.00
0.07	201.00	146.00	155.00
0.08	193.00	141.00	138.00
0.09	188.00	137.00	130.00
0.10	184.00	134.00	123.00
Sky			
0.03	233.00	326.00	413.00
0.04	210.00	290.00	367.00
0.05	164.00	231.00	295.00
0.06	131.00	183.00	231.00
0.07	108.00	150.00	193.00
0.08	92.00	132.00	165.00
0.09	90.00	123.00	156.00
0.10	88.00	120.00	153.00
Lake			
0.03	803.00	771.00	803.00
0.04	638.00	632.00	668.00
0.05	426.00	433.00	462.00
0.06	283.00	289.00	309.00
0.07	202.00	218.00	232.00
0.08	194.00	194.00	204.00
0.09	189.00	189.00	189.00
0.10	185.00	185.00	185.00

B.5 Results from testing the smoothing metric

Table B.16: Results from simulating the *Gaussian filter-based* metric using natural images with different levels of Gaussian noise.

Excessive blurring			
Gaussian noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	4411531	4346250	4330319
0.02	6016291	5896842	5856248
0.03	7161986	7046300	6959717
0.04	8107044	7970917	7824750
0.05	8862740	8728193	8544282
0.06	9552712	9361418	9189918
0.07	10176095	9991604	9774152
0.08	10679183	10517031	10283809
0.09	11214735	10986847	10747650
0.10	11614524	11428337	11171034
Sky			
0.01	3261581	3278474	3286061
0.02	4557912	4597591	4516015
0.03	5538728	5616959	5372169
0.04	6339357	6451129	6054879
0.05	6955050	7124335	6602802
0.06	7529692	7745426	7091914
0.07	8019252	8287573	7553730
0.08	8454760	8738771	7940092
0.09	8788080	9148644	8309588
0.10	9205300	9515678	8622354
Lake			
0.01	6989491	6931811	6204916
0.02	8809067	8729109	7813162
0.03	10198806	10074879	9048099
0.04	11286756	11147762	10053222
0.05	12276802	12114551	10934290
0.06	13113605	12990291	11746921
0.07	13859511	13667273	12427343
0.08	14570862	14386053	13099495
0.09	15194594	14984203	13641295
0.10	15717802	15550433	14228083

Table B.17: Results from simulating the *Gaussian filter-based* metric using natural images with different levels of salt & pepper noise.

Excessive blurring			
Salt & pepper noise			
Noise density	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	1435314	1289844	1342313
0.02	1819109	1706971	1781164
0.03	2264569	2102495	2186955
0.04	2633973	2514342	2622842
0.05	3048413	2941422	2966962
0.06	3422714	3342083	3332053
0.07	3810177	3712710	3781252
0.08	4157331	4090998	4134698
0.09	4543032	4480905	4528894
0.10	4937278	4790857	4848368
Sky			
0.01	543842	513886	572058
0.02	898133	864923	909618
0.03	1223562	1210752	1254392
0.04	1570421	1520496	1601571
0.05	1870908	1824934	1915916
0.06	2225211	2137050	2225034
0.07	2480202	2462818	2590452
0.08	2776715	2766016	2887157
0.09	3083068	3072877	3212452
0.10	3354285	3320670	3468970
Lake			
0.01	3805560	3842950	3632645
0.02	4337434	4386844	4179989
0.03	4825799	4863453	4717384
0.04	5366380	5368405	5257254
0.05	5862867	5864689	5787975
0.06	6331658	6366581	6325987
0.07	6827649	6849659	6849875
0.08	7342704	7425450	7333615
0.09	7815611	7879283	7792017
0.10	8333864	8338936	8355868

TABLE B.18: Results from simulating the *Gaussian filter-based* metric using natural images with different levels of speckle noise.

Excessive blurring			
Speckle noise			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.01	2696633	1758392	1594703
0.02	3567762	2285293	2010787
0.03	4270082	2709036	2360136
0.04	4849784	3070702	2647632
0.05	5349207	3395705	2919494
0.06	5793872	3690925	3170280
0.07	6199283	3964596	3388391
0.08	6564687	4208619	3599151
0.09	6929044	4459570	3797350
0.10	7229341	4688599	3986374
Sky			
0.01	1232655	1836012	2637070
0.02	1726408	2564272	3693473
0.03	2094707	3124588	4489529
0.04	2418491	3606987	5114855
0.05	2697567	4015237	5660785
0.06	2937405	4407547	6113300
0.07	3191411	4755695	6514098
0.08	3397541	5083001	6884177
0.09	3599883	5376205	7210858
0.10	3795695	5670864	7495271
Lake			
0.01	4998202	5103418	4550159
0.02	5863339	5928901	5293799
0.03	6525663	6589182	5880903
0.04	7096494	7174560	6415837
0.05	7594657	7666355	6873351
0.06	8042664	8135537	7291878
0.07	8473655	8549102	7680391
0.08	8850803	8963755	8057763
0.09	9215027	9329943	8421559
0.10	9567923	9689349	8711611

TABLE B.19: Results from simulating the *Gaussian filter-based* metric using natural images with different levels of blurring.

Excessive blurring			
Blurring			
Noise variance	**Red channel**	**Green channel**	**Blue channel**
Lena			
0.30	991977	846445	926164
0.40	877667	746640	810104
0.50	687697	574813	616467
0.60	547011	448875	475441
0.70	461245	371455	390428
0.80	413222	328775	343243
0.90	385783	304548	317027
1.00	369580	290662	301908
Sky			
0.30	187159	145594	192373
0.40	180990	142664	188059
0.50	124182	115340	146882
0.60	83489	80762	107940
0.70	60405	65095	88053
0.80	49949	57750	79031
0.90	45265	52784	74026
1.00	43499	52011	72344
Lake			
0.30	3249225	3289309	3064694
0.40	2733260	2771852	2573886
0.50	1945478	1978079	1839343
0.60	1376414	1400726	1304416
0.70	1046835	1066274	997522
0.80	873932	889480	835964
0.90	784711	797980	752541
1.00	741017	752734	711915

Bibliography

[1] Eirik Tørud Nordeng. Video metric measurements in an FPGA for use in objective no-reference video quality analysis, 2012.

[2] Donald G. Bailey. *Design for Embedded Image Processing on FPGAs.* John Wiley & Sons, May 2011. ISBN 9780470828502.

[3] Altera Corporation. Embedded multipliers in cyclone II devices, February 2007. URL `http://www.altera.com/literature/hb/cyc2/cyc2_cii51012.pdf`.

[4] Katherine Compton and Scott Hauck. Reconfigurable computing: a survey of systems and software. *ACM Comput. Surv.*, 34(2):171210, June 2002. ISSN 0360-0300. doi: 10.1145/508352.508353. URL `http://doi.acm.org/10.1145/508352.508353`.

[5] Keith Jack. *Video Demystified: A Handbook for the Digital Engineer.* Elsevier, April 2011. ISBN 9780080553955.

[6] Rafael C Gonzalez and Richard E Woods. *Digital image processing.* Prentice Hall, 2008. ISBN 9780131687288.

[7] G.L. Bates and S. Nooshabadi. FPGA implementation of a median filter. In *, Proceedings of IEEE TENCON '97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications*, volume 2, pages 437–440 vol.2, 1997. doi: 10.1109/TENCON.1997.648210.

[8] Stefan Winkler. *Digital Video Quality: Vision Models and Metrics.* Wiley, March 2005. ISBN 9780470024041.

[9] R. Ferzli and L.J. Karam. A no-reference objective image sharpness metric based on just-noticeable blur and probability summation. In *IEEE International Conference on Image Processing, 2007. ICIP 2007*, volume 3, pages III – 445–III – 448, 2007. doi: 10.1109/ICIP.2007.4379342.

[10] R. Ferzli and L.J. Karam. A no-reference objective image sharpness metric based on the notion of just noticeable blur (JNB). *IEEE Transactions on Image Processing*, 18(4):717–728, 2009. ISSN 1057-7149. doi: 10.1109/TIP.2008.2011760.

[11] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahimi. A no-reference perceptual blur metric. In *2002 International Conference on Image Processing. 2002. Proceedings*, volume 3, pages III–57 – III–60 vol.3, 2002. doi: 10.1109/ICIP.2002.1038902.

[12] M. Masry, S.S. Hemami, and Y. Sermadevi. A scalable wavelet-based video distortion metric and applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 16(2):260–273, 2006. ISSN 1051-8215. doi: 10.1109/TCSVT.2005.861946.

[13] N.D. Narvekar and L.J. Karam. A no-reference perceptual image sharpness metric based on a cumulative probability of blur detection. In *International Workshop on Quality of Multimedia Experience, 2009. QoMEx 2009*, pages 87–91, 2009. doi: 10.1109/QOMEX.2009.5246972.

[14] Xiang Zhu and P. Milanfar. A no-reference sharpness metric sensitive to blur and noise. In *International Workshop on Quality of Multimedia Experience, 2009. QoMEx 2009*, pages 64–69, 2009. doi: 10.1109/QOMEX.2009.5246976.

[15] R. Dosselmann and Xue Dong Yang. A prototype no-reference video quality system. In *Fourth Canadian Conference on Computer and Robot Vision, 2007. CRV '07*, pages 411–417, 2007. doi: 10.1109/CRV.2007.6.

[16] J.S. Lee and K. Hoppel. Noise modeling and estimation of remotely-sensed images. In *Geoscience and Remote Sensing Symposium, 1989. IGARSS'89. 12th Canadian Symposium on Remote Sensing., 1989 International*, volume 2, pages 1005–1008, 1989. doi: 10.1109/IGARSS.1989.579061.

[17] J. Caviedes and Sabri Gurbuz. No-reference sharpness metric based on local edge kurtosis. In *2002 International Conference on Image Processing. 2002.*

Proceedings, volume 3, pages III–53–III–56 vol.3, 2002. doi: 10.1109/ICIP.2002.1038901.

[18] Yongfeng Wang, Haiqing Du, Jingtao Xu, and Yong Liu. A no-reference perceptual blur metric based on complex edge analysis. In *2012 3rd IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pages 487–491, 2012. doi: 10.1109/ICNIDC.2012.6418801.

[19] Alan Burns and Andrew J. Wellings. *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*. Addison-Wesley, April 2009. ISBN 9780321417459.

[20] National Instruments. How to identify common video defects with the NI analog video analyzer, 2011.

[21] Jørgen Linnerud. Personal communication via e-mail: Known analog video system issues, November 2012.

[22] Alan C. Bovik. *Handbook of Image and Video Processing*. Academic Press, July 2010. ISBN 9780080533612.

[23] National Instruments. Picture quality analysis: Real-time measurements for objective video quality, June 2012.

[24] Analog Devices. AD9388A, 2010. URL `http://www.analog.com/static/imported-files/data_sheets/AD9388A.pdf`. Rev. F.

[25] Altera Corporation. RAM-Based shift register (ALTSHIFT_TAPS) megafunction - user guide, 2013.