UPPSALA UNIVERSITET

UPTEC IT 17025
Examensarbete 30 hp
November 2017

Denoising and renoising of video for compression

Anders Derk Gärdenäs

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Denoising and renoising of video for compression

Anders Derk Gärdenäs

Videos contain increasingly more data due to increased resolutions. Codecs are further developed and improved to reduce the amount of data in videos. One difficulty with video encoding is noise handling, it's expensive to store noise and the final result is not always aesthetically pleasing. In this thesis project an algorithm is developed and presented which improves the visual quality while reducing the bit-rate of the video, by improved management of noise.

The aim of the algorithm is to store noise information in a specific noise parameter instead of mixing the noise with the visual information. The algorithm was developed to be part of the modern codec JEM, a successor of the h.264 and h.265 codecs. The algorithm can be summarized in the following steps: the first step is to identify how much noise there is in the video, which is done with a temporal noise identification algorithm. The noise identification is done at the start of the encoding process. The second step is to remove noise from the video with a denoising algorithm, this is done during the encoding processes. The third and final step is reapplication of the noise, this is done using the noise parameters computed in step one. The third step is done during the decoding phase. The result was evaluated in a subjective survey consisting of five people evaluating 27 different versions of three videos.

The result of the subjective survey shows a consistently improved visual quality resulting from the proposed technique, achieving an improved score from 3.35 to 3.6 on average on a subjective 1-5 scale where 5 is the best score. Furthermore, the bit-rate was significantly reduced by denoising. Bit-rate reduction is particularly high in high-quality videos, where the average reduction of as much as 49% is achieved. Another finding of this thesis is that the same video quality can be achieved using 2.7% less data by using a denoising tool as part of the video encoder. In conclusion, it is possible to improve video quality while reducing the bit-rate using the proposed method.

Handledare: Per Wennersten
Ämnesgranskare: Natasa Sladoje
Examinator: Lars-Åke Nordén
UPTEC IT 17025
Tryckt av: Reprocentralen ITC

Sammanfattning

Mängden data i videoklipp växer i takt med att upplösningen blir större. Kodeks vidareutvecklas och förbättras för att minska mängden data i videoklipp. En svårighet med videoklipp kodning är brushantering, det kräver mycket data att spara brus och det visuella resultatet är inte alltid bra. I denna rapport utvecklades en algoritm som förbättrar videokvalitén och samtidigt minskar bitraten i videoklippet, detta genom att hantera brus bättre.

Målet med algoritmen är att spara brusinformation i en specifik brusparameter istället för att blanda brusdata med video data. Algoritmen är utvecklad för att vara del av kodeken JEM, en efterföljare av kodekarna h.264 och h.265. Algoritmen kan sammanfattas med följande steg: det första steget är att identifiera mängden brus i videoklippet, detta görs med hjälp av en temporal brusidentifierings algoritm. Brusidentifikationen sker innan kodningen av videoklippet. Det andra steget är att ta bort brus från videoklippet med en brusbortagningsalgoritm, brusborttagningen sker under kodningsprocessen. Det tredje och sista steget är återapplicering av brus, detta steg sker med hjälp av brusparametrarna uträknade i steg ett. Sista steget sker under avkodningsprocessen. Resultatet är evaluerat i en subjektiv undersökning där fem personer som evaluerade 27 olika versioner av tre videoklipp.

Resultatet av den subjektiva undersökningen visar att den utvecklade tekniken förbättrar den visuella kvalitén. Med hjälp av brusbortagning och återapplicering av brus förbättrades den genomsnittliga subjektiva poängen från 3.35 till 3.6 på en 1-5 skala. Dessutom minskade bitraten signifikant, i genomsnitt 49% för videos med hög kvalité. I denna rapport visades också att samma visuella kvalité kan nås med 2.7% mindre data genom att använda ett brusbortagningsverktyg i kodningsprocessen. Sammanfattningsvis är det möjligt att förbättra videokvaliteten samtidigt som bitraten minskas med den föreslagna metoden.

Contents

List of Figures

List of Tables

Acronyms

BD-rate Bjøntegaard-delta rate.

CRF Camera Respons Function.

NLF Noise Level Function.

QP Quantization Parameter.

1. Introduction

1.1 Motivation

Videos contain more and more data due to increased resolutions. In order to cope with large amounts of data, new and better video encoders and decoders are made [Eri17]. There are many ways to reduce the amount of data in a video. Some Algorithms, called lossless compression, preserve the video quality. Other algorithms called lossy compressing algorithms lose some information when compressing. The idea of lossy compression algorithms is to remove information which the viewer does not see or care about, however what some can see or care about can be subjective and it can therefore be hard to implement a good lossy compression algorithm [NG96].

1.2 Problem formulation

Noise in videos can come from different sources. Cameras introduce noise due to their design, but noise can also be added by video creators for artistic effect or to hide flaws in digital effects [Bro13]. This creates a problem for video encoders since noise is very expensive to code due to its randomness. Moreover, the noise cannot simply be removed, because it may be desired [OLK09].

1.3 Aims and hypotheses

The aim of this project is to develop a lossy video compression algorithm, which in the encoding phase extracts the noise characteristic of the video, and removes the noise from the video. In the decoding phase the noise characteristic will be used to reproduce the noise; the noise will be subjectively similar to the original video. The specific aims are:

- To remove noise from a video in such a way that the information is preserved and that the video takes less memory to store.
- To identify noise characteristic of a video systematically and store the noise characteristic in a small amount of memory.
- To reproduce noise to a video resulting in subjectively good quality.

Furthermore, the following hypothesis will be tested.

- Can a modern encoder be improved with denoising techniques in a way where the video quality is improved relative to the amount of data needed to store the video?

2. Background

2.1 Related work

Previous work within the field has focused on implementations which remove and then reapply film grain, noise from analog cameras. Thomson co(2004) implemented and patented Film Grain Technology (FGT) [LAG13], a denoising and renoising technology with the aim to save space in the encoded video. FGT focuses on saving memory by storing parameters of the film grain in the encoded video, for example the intensity of the grain, the size and the color. The film grain parameters is used to recreate similar film grain to the grain of the pre-encoded video [LAG13]. FGT was set as a mandatory standard in HD DVD-Video by DVD Forum(2005) [For05]. In 2007 a new technique was presented by Byung Tae Oh, Shaw-min Lei and C.-C. Jay Ku for IEEE [OLK09]. A very similar approach was used in this project, although this project focuses on digital camera noise and not analog camera noise. The technique presented in [OLK09] consists of three steps, first a denoising step, secondly retrieving the noise characteristic and lastly reapplying the noise. The first step, denoising, is divided into two parts: detecting smooth areas with an edge detecting technique and denoising smooth areas with a temporal denoising algorithm. The second step, retrieving the noise uses an autoregressive model which considers factors like the spatial power spectrum density, the noise probability density and the crosscolor correlation to module the film grain. The last step is constructing the final image with the help of the autoregressive model constructed in the previous step. Finally, the paper concludes that the module can significantly improve bit-rate without affecting the visual quality [OLK09].

2.2 Image sensors

Charge Coupled Device (CCD) and Complementary Metal-Oxide Semiconductor (CMOS) sensors are the most common devices to capture light in digital cameras, and they are the devices which this project focuses on. In general is the quality of a CCD sensor better and the amount of noise lower compared to a CMOS sensors, however CCD sensor are more expensive [LGLS08]. A CCD photon detector consists of a thin silicon layer divided into a geometrical array of up to millions of light sensitive regions. Every region captures and stores image information in the form of electrical charge that varies with intensity of the light captured. The electrical charge is then transported to be converted to a digital signal and stored as pixel values in an image. The location of the pixel in the image corresponds to the location of the region where the light was captured on the CCD [SFD10]. CMOS sensors work similar to the CCD, the first step of the CMOS sensors is to collect light information and convert it to electrons in a similar fashion as the CCD. Unlike the CCD the electrons are directly converted to a digital signal within the CMOS sensor. The CMOS sensors only capture a row at a time compared to the CCD which capture the entire image [LGLS08].

2.2.1 Camera response function

To understand how a digital camera is affected by noise it's necessary to understand how the digital camera translates irradiance to different luminance values. Luminance is a photometric measure of the luminous intensity per unit area of light traveling in a given direction. The Camera Respons Function (CRF) is the function describing which number of photons translates to which value of luminance in the captured image or video. The CRF will not be used directly in this thesis, however the CRF indirectly effects some of processes used in this thesis and it is therefore necessary to know how it alters the video. The CRF is a nonlinear function which depends on many parameters, for example lens fall-off and the sensitivity of the detector in the camera. The CRF can also be altered to better match different visualization technologies such as gamma correction [CLYY12].

2.3 Noise

Noise in videos can come from many different sources. Cameras have many sources of noise, but noise can also be added by content creators for artistic effect or to hide flaws in digital effects [BKE^{+}95]. There are three different main sources of noise in the digital camera, *Shot noise*, *Dark current* noise and *Readout Noise*. The following sections will describe these noise sources and what affects them.

2.3.1 Shot noise

Image sensors in the digital cameras are capturing light and translating it to an image. Light is made out of photons so to capture luminance, is to count the number of photons captured. The more photons, the brighter the image. However, the number of captured photons are not constant over time due to the discrete nature of photons, this fluctuation is called shot noise. Shot noise has Poisson distribution which has a standard deviation of $\sqrt{\lambda}$ where λ is the luminance. The square root growth of Shot noise compared to the luminance means that the relative amount of noise will shrink the stronger the luminance. Figure 2.1 visualizes varying degrees of shot noise, the stronger the intensity the less visible the shot noise is [WS98].

Figure 2.1. A simulation of shot noise. The number of absorbed photons per pixel increases from left to right and from upper row to bottom row (0.001 to 100 000 photons per pixel). The more photons the less relative strength of the shot noise.
The figure is retrieved from `https://commons.wikimedia.org/wiki/File:Photon-noise.jpg`

2.3.2 Dark current noise

Dark current is generated by imperfections in the silicon substrate on the image sensor. The imperfections of the silicon cause electric invariance which creates paths for valence electrons to move and alter the signal representing the pixel. The dark current is somewhat predictable and its effect can therefore be removed. However, there is some noise in the dark current, called dark current noise. Dark current noise is Poisson distributed relative to the amount of dark current. The amount of dark current is affected by the amount of heat energy, with more energy more electrons will move further increasing the dark current and thus increasing the amount of dark current noise. The amount of dark current can be reduced by cooling the image sensor, which reduces the amount of dark current noise. [Kod01].

2.3.3 Readout Noise

Readout Noise also called amplifier noise is the noise created when electronic charges from the image sensors are converted to measurable voltage. Readout is depended on the quality of the hardware in the camera and not dependent on the electronic charge of the signal. The readout noise is therefore relatively stronger noise source in low signal levels whereas for high signal levels the relative noise is low [HK94].

2.3.4 Total noise of the digital camera

The amount of noise in the CCD or CMOS sensor may vary, however all the described noise models affects both sensors types. The noise of the digital camera has many sources, with varying amount of effect on the final result. Additionally, during the conversion from analog to digital signal some data is lost in the quantization process. Of these noise sources shot noise is the only noise type in which strength varies throughout the image, this is because shot noise grows with the luminance [HK94]. The following model derived from the noise model proposed by [LFSK06] describes the total noise of the digital camera.

$$I = CRF(n_s + n_c) + n_q, \tag{2.1}$$

where I is the total noise, $CRF(\cdot)$ is the camera response function, n_s represents the noise depended the on the luminance, the shot noise, n_c represents the static noise affected by the CRF, the dark current noise, n_q represents noise independent of CRF, which is the quantization and readout noise.

2.3.5 Noise level function

The total noise model described in section 2.3.4 can be interpreted as function dependent on the luminance. This function is called the Noise Level Function (NLF) and describes the expected amount of noise at a given luminance. NLF consist of two terms, one constant and one dependent on the luminance. The term dependent on the luminance originates from shot noise, which is linear function of the square root of the luminance. Omitting all effects of the CRF the NLF of the digital camera can be described with the following formula [LFSK06].

$$NLF(L) = k * \sqrt{L} + m, \tag{2.2}$$

where the NLF relates to the square root of luminance L, k is the strength of the shot noise and m is the strength of all the static noise. In theory can the NLF be extracted from an image or a video, however there are some limitations. [KOS10] showed that the NLF is greatly altered by the CRF. Because of the many irregularities in the CRF [KOS10] it can be complicated to compute the exact CRF from an image and it's therefore non trivial to identity the exact NFL of an image. Instead of detecting the NLF, the NLF altered by the CRF can be approximated utilizing equation 2.3.

$$CRF(NLF(L)) = CRF(k * \sqrt{L}) + m_{crf}, \tag{2.3}$$

where $CRF(\cdot)$ is the camera response function, m_{crf} is all the static noise adjusted for the CRF. In this thesis is it not necessary to know the exact NLF, rather the NLF adjusted to the CRF is used, the CRF(NLF). How the CRF(NLF) is estimated and used will be described in in the coming sections. Any future mention of the NLF will be assumed to be CRF(NLF).

2.3.6 Generating noise

The total noise in the digital camera, described in Section **??** is both spatially and temporally independent. Video noise is most commonly modeled by Gaussian random noise as described in [Bar13]. However as described in section **??** the noise is dependent on the luminance and therefore the the reapplied noise should be determined utilizing NLF. In each region where noise is applied its luminance value should be identified and used in the NLF estimated from the original video to identify its Noise level. The metric used to measure the NLF is the *noise level*. In this thesis a *noise level* of *N* is defined as Gaussian noise with a standard deviation of *N* i.e., if a video has a *noise level* three then the noise in the video has Gaussian distribution with a standard deviation of three. The Peak Signal to Noise Ration (PSNR) is used as a metric in Section 2.6.1, for comparison of 8-bit images. As a reference, a noise level 1 corresponds to PSNR of 48.1, Pseudo-code for converting *noise level* to PSNR can be found in Appendix E.

2.4 Denoising

The process of denoising includes identifying noise and then removing it. Identifying noise can be hard due to the randomness of noise, an algorithm which tries to remove all noise might accidentally remove some information resulting in loss of video quality. If the denoising algorithm tries to preserve all information it might be inefficient at identifying noise, resulting in incomplete denoising [CEPY05].

2.4.1 Linear and nonlinear filtering

In denoising it is common to use information from adjacent pixels to estimate the denoised intensity value of a pixel. Mean filtering is an example of such an algorithm. The mean filtering algorithm operates by computing the mean value of a pixel and all the pixels around it and uses the computed value as the denoised value. The idea of mean filtering is that if all the adjacent pixels had the same pre-noise intensity then the pre-noise color can be estimated by calculating the mean of the noisy values. However, mean filtering has some limitations, if the pixel to denoise is adjacent to an edge of a different intensity the edge will be distorted. Instead a nonlinear filter is more suitable. An example of a nonlinear filter is the Median filter which operates similar to Mean filtering except it uses a median function instead of a mean function. Another example of a nonlinear filter is a filter where each adjacent pixel has a weighted impact of the final denoised value. The closer the intensity value of the adjacent pixel is to the intensity value of the pixel to be denoised, the higher the weight of that pixel is and thus its final impact [Buc70]. In this thesis, an advanced weighted nonlinear filter will be used to denoise videos, being more efficient than the linear alternatives [Buc70].

2.4.2 Domain of filters

A video can be interpreted as a three-dimensional signal, where two dimensions represent the spatial location, the x and y coordinate of a pixel in a given frame of video and the third dimension represents the temporal location, the frame index of a video. Different filtering techniques use different domains of the video, examples are spatial filtering and temporal filtering.

Spatial filtering
Spatial filtering techniques operate in the spatial domain of a video, meaning that they only operate on one frame at a time. Spatial filtering of a video and filtering of an image are therefore similar and techniques used for images filtering can be applied in spatial filtering.

Temporal filtering
Temporal filtering techniques use consecutive frames to filter. The idea is that a video will not change much between consecutive frames whereas noise does. By looking at the difference of the two frames the noise can be detected and removed. Movement in video can reduce the performance of temporal filtering, therefore motion vectors are sometimes used to counter movement [BKE+95].

2.4.3 Wavelet Filtering

Wavelet-based filters rely on the wavelet transform on the video signal to decompose it into components of different frequency intervals. Applying the assumption that noise frequencies have a low amplitude, the different frequency components can be limited by a threshold and thus make it possible to remove the noise [SM99].

2.5 Video compression

There are a wide range of different video compression techniques [Ric04]. Without compression the bit-rate of the video will double if the frame rate or resolution doubles [CPW11], however with compression better bit-rate to frame rate and resolution ratios can be achieved. The following section will present the two video compression techniques, *Interframe Video Coding* and *Quantization*. *Interframe Video Coding* has an important role of video compression, however the performance of the technique is effected by noise [OLK09].

Furthermore, *motion vector search*, a sub technique of *Interframe Video Coding*, will also be used in noise detection. *Quantization* is important for two reasons in this thesis. First it removes some noise in the compression processes [Ric17], secondly *Quantization* is used to control how strong the video compression is. The video codec used in this project is the Joint Exploration Model (JEM) codec which is based on the High Efficiency Video Coding (HEVC) standard developed by the Joint Video Exploration Team (JVET) [HI17]. This thesis was done together with Ericsson research and together we chose to use the JEM codec, however there is not a technical reason why JEM was chosen except it being modern.

2.5.1 Interframe Video Coding

Reusing data between frames is an essential part of efficient video coding. The idea is that there won't be much change between two following frames, and much of the difference is due to movement rather than new items in the frame, so most parts of the old frame can be reused in the new frame. This is achieved with a motion vector search algorithm. The motion vector search operates between two frames, an Intra frame (I-Frame) and a predicted frame (P-Frame), where the P-Frame is a later stage of the video than the I-Frame. The P-Frame is divided into a grid, where each block of the grid is an N x N pixel block. The next step is to predict how each block has moved between the I-Frame and the P-Frame, so for each block in the P-Frame the task is to find the best matching block in a search region of the I-Frame. This is visualized in Figure 2.2. A motion vector is computed utilizing the positions of a current block and its best match in the I-frame. The match is computed by computing the sum of the absolute differences between the two blocks, the lower difference the better match. The initial frame of a video will be an I-Frame, however more I-Frames exist as resynchronization points throughout the video [HP12]. Furthermore, Bi-predictive frames (B-Frame) can be used; the B-Frame operates much like the P-Frame except the B-Frame also uses motion vector search with the following frame [HP12]. Noise can significantly decrease the performance of the motion vector search, because the noise is temporally independent resulting in inaccurate block compression which hampers the accuracy of the motion prediction [OLK09].

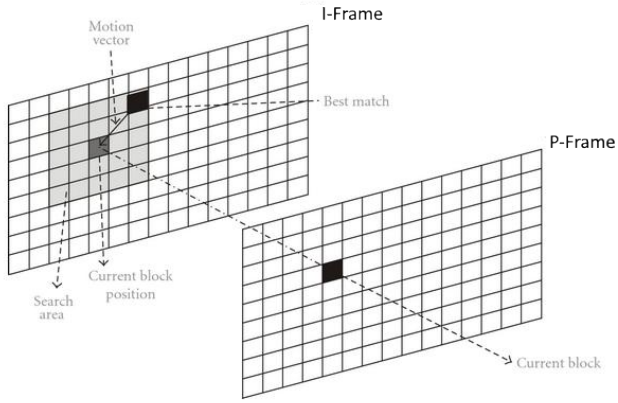

Figure 2.2. Motion vector search. A best match for the block in the P-Frame is found among a set of blocks from the I-Frame and the difference in locations is saved as a motion vector. The figure is an adapted version of the figure at:
`https://www.hindawi.com/journals/ijrc/2012/473725/fig1/`

2.5.2 Quantization

Quantization is used to further reduce the amount of data within a video. The goal is to reduce some frequency components of the video signal which the human eye can hardly detect. For every block the data is converted into the frequency domain utilizing the Discrete Cosine Transform. The frequency information is stored in a matrix M where the rows and columns represent the frequencies in the directions of the x and y axis, respectively. In the next step M is divided element-wise by the quantization matrix Q, which can be defined and adjusted to the particular needs. After the division the values are quantized to discrete values, this is the part were data is saved. Because of the quantization some values may be rounded down to zero, thus losing any information they previously had. During the decoding the inverse transform of Q is used to restore the values [Ric17]. Quantization Parameter (QP) is used to control the quality of the quantization. QP determines the quantization matrix used, the higher the QP value the fewer frequency components will be saved. QP is numbered after its strength: the higher the number, the stronger the compression i.e., QP22 results in a better video quality than QP37, but QP37 compress more data [WK08].

2.6 Measuring video quality

Measuring video quality can be done both objectively and subjectively. This section will describe some of the methods used to measure the quality of the video. To measure video quality objectivity Peak Signal-to-Noise Ratio and

Structural Similarity Index Measure are used. These two methods have different focuses: where Peak Signal-to-noise ratio directly measures the difference between the two images, SSIM tries to take human perceived image quality into account.

2.6.1 Peak Signal-to-noise ratio

Peak Signal to Noise Ratio (PSNR) is an evaluation method to measure loss of video quality. The PSNR measures average of the squared pixel-wise differences between two images compared to the maximum possible difference i.e., the maximal possible pixel value in the image. The PSNR metric is expresses in units of decibels [dB]. The following formula calculates the PSNR of the two images f and g:

$$PSNR(f,g) = 10 * log_{10}(\frac{P^2}{MSE(f,g)}) \tag{2.4}$$

$$MSE(f,g) = \frac{1}{H*W} \sum_{i=1}^{H} \sum_{j=1}^{W} (f_{ij} - g_{ij})^2 \tag{2.5}$$

where P is the peak pixel value of the intensity space used, H is the height and W the width of the images f and g. A smaller MSE(f,g) indicates a smaller difference between f and g, hence the more similar f and g are the higher the *PSNR(f,g)* value is [HZ10]. If image g is a compressed image and f is the same image before the compression *PSNR(f,g)* can be used to measure the loss of image quality in the compression. If *PSNR(f,g)* is a high, the loss of image quality in the compression is low.

2.6.2 Structural similarity index measure

Structural Similarity Index Measure (SSIM) is a quality metric to measure similarity between two images, not only based on raw image difference but also the quality perception of human visual system. The SSIM metric is based on three different factors, the loss of intensity, luminance distortion and contrast distortion. The SSIM score goes from zero to one where one means that the two images are identical [HZ10].

2.6.3 Bjøntegaard-delta

Bjøntegaard-delta (BD) model is used to estimate the efficiency between two codecs based on PSNR and bit-rate measurements. BD uses PSNR measurements of a video at multiple bit-rate levels to construct an estimation for any given bit-rate, enabling a direct comparison between two codecs for a given video and bit-rate range. This is done with two rate-distortion curves generated by the PSNR/Bit-rate measurement points. The actual BD is computed based on the difference between the two rate-distortion curves. The Bjøntegaard-delta rate (BD-rate), is the mean bit-rate difference in percent for the same PSNR value. For example, if video codec A has a BD-rate of -2% compared to video codec B, that means that A requires 2% less data for the same video quality compared to B. BD-Rate can therefore be used to estimate how much better or worse a codec is compared to another codec both with respect to quality and the bit-rate [Bjo01].

3. Materials and Methods

3.1 Overview

The following section gives a short description of the algorithm for the entire project. See Figure 3.1 for a summary and example frames.

1. The first step is to identify the noise level function (NLF) of the video, NLF is used to measure the amount of noise present in the video. Two different NLF identification methods were implemented: One temporal and one spatial. The two NLF identification methods were evaluated and the temporal method scored the best in the evaluation and was therefore finally used.
2. The following step is to denoise the video. Three different denoising algorithms were evaluated; *MCSpudsmod* a denosing tool which is part of the video post-production tool *AviSynth* [RG03], *Owdenoise* and *Hqdn3d* both part of the multimedia framework *FFmpeg* [Bel16]. *MC-Spudsmod* scored the best in the evaluation and is the denoising algorithm selected to be used in the final procedure.
3. The third step was to encode the video and then decode it. This was done using the JEM codec version 4.1.
4. The final step is to reapply the noise to the video. The amount of noise added is given by the NLF which was computed in the first step.

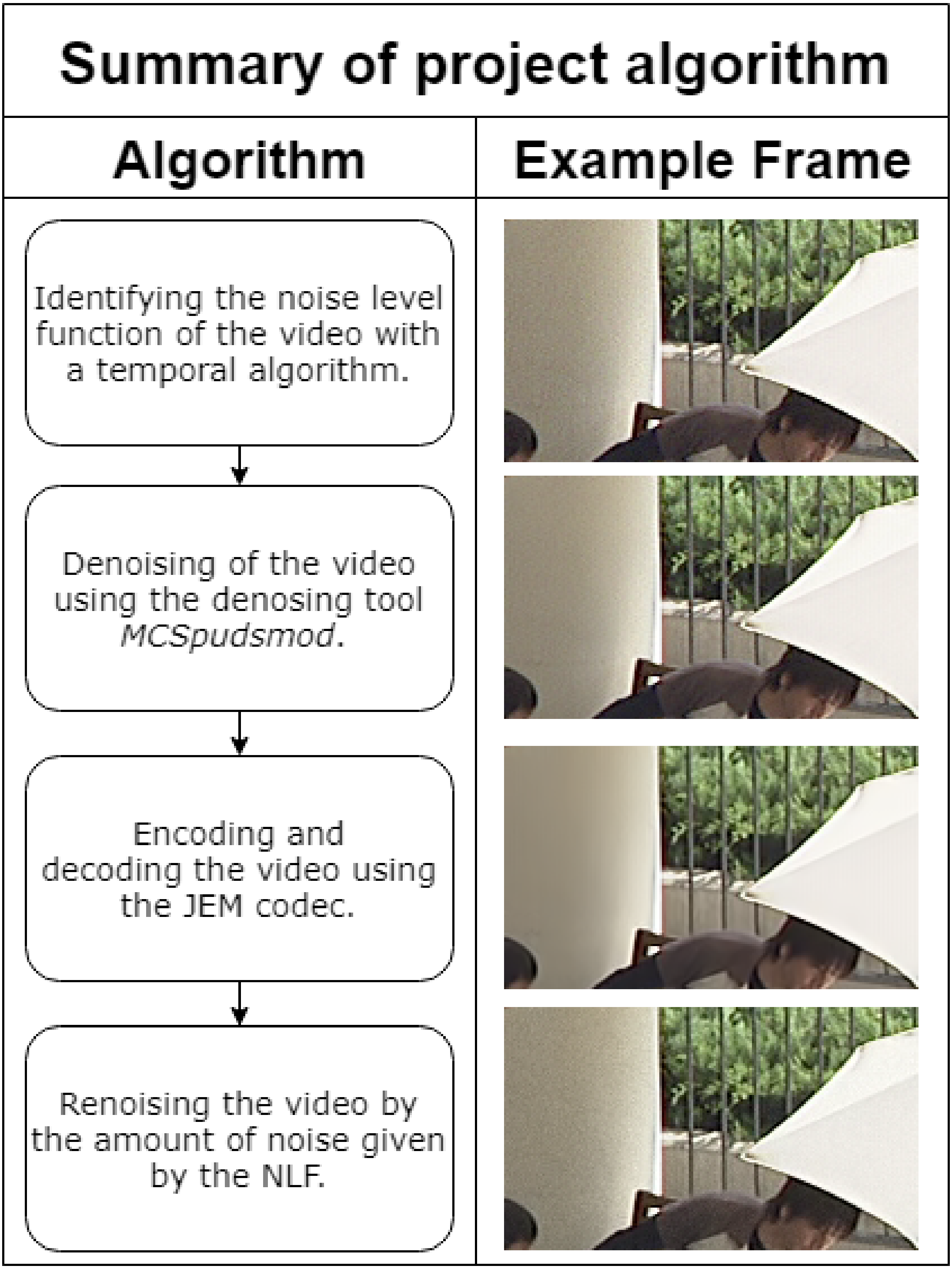

Figure 3.1. Summary of project algorithm. The noise in the frames can for example be observed in the pillar to the left of each image.

3.1.1 Video test suite

A test suite of 24 different videos was used to evaluate the different stages of the algorithm. The source of the videos is Joint Video Exploration Team (JVET) test cases [Jac11]. The videos have many different quality settings such as spatial resolution ranging from 416x240 to 4096x2160, frame rate from 20 to 100 frames per second and bit depth ranging from 8 to 10bit. The videos are divided into 5 different groups depending on the spatial resolution. The groups are named from A to E where A consists of the videos of highest video quality and E are the videos of lowest quality. All the videos in the test suite were captured using a digital camera, except for two of them which were computer generated [OS13].

3.2 Noise parameters

To further widen the understanding of the noise in the video a NLF analyzing program was created. The aim of the program was to identify the NLF of a video. Two different approaches where tested to compute the NLF, one temporal and one spatial.

3.2.1 Spatial noise level function identification

The spatial NLF identifying algorithm is based on [CB13]. The idea is that there will be multiple homogeneous regions within the frame and these regions can be used to identify noise. A homogeneous region has little variance except for noise, therefore the noise can be estimated by measuring the amount of variance within the region. We assume 10% of the regions are homogeneous, find the 10% of regions with the least amount of variation for every luminance level and use these regions as computational ground for the NLF. More specifically, the algorithm is described by the following steps: For every pixel P_{xy} in the frame create a block with the pixel at its center and width and height of *(2*r + 1)*, where r determines the radius of the block and x and y are the pixel's vertical and horizontal location in the frame. Then the standard deviation and the luminance of the block are computed; the luminance I_{xy} is the mean value of all the pixels in the block and sd_{xy} is the standard deviation of the block. Then for every block the standard deviation values sd_{xy} are grouped by their luminance value I_{xy} into the array of sets *Deviation[]* with the following expression:

$$Deviation[i] = \{sd[x,y] | i = I_{xy}\} \tag{3.1}$$

Lastly for every luminance level, the mean of the 10% smallest *Deviation[i]* are used as the noise level at that luminance level. The NLF is now estimated, for every luminance level there is a corresponding noise level. The region with the least amount of variation is used because they are the most likely to represent a region with only noise. If a region has a lot of variation then it's likely not homogeneous region, however if the amount of variation is low then the little variation that exists is more likely caused by noise.

3.2.2 Temporal noise level function identification

Temporal estimation of NLF is the second approach to identify the NLF. In this case the NLF is computed by calculating the differences between two consecutive frames. The approach is based on [KOS10], and the idea is that if no movement has occurred between two frames then the difference between the two frames will be noise. The temporal NLF identification algorithm can be described by the following steps: The NLF is calculated for every frame f_n where f_n is any frame before the last frame in the video. For every frame the following difference is computed:

$$D[x,y] = |f_n[x,y] - f_{n+1}[x,y]| \tag{3.2}$$

D contains the absolute pixel-wise difference at every position. Then all the values of D are grouped according to their intensity values f_n[x,y] into the array of sets *Deviation[]* with the following expression:

$$Deviation[i] = \{D[x,y] | i = f_n[x,y]\} \tag{3.3}$$

Lastly every *Deviation* value is assigned a NLF by computing the mean:

$$NLF[i] = mean(Deviation[i]) \tag{3.4}$$

Assuming a static video, *NLF[]* will represent the noise level function of the video. The previous calculation assumed a static video, however this is not always the case, rather some movement in the video should be expected. To compensate for movement, motion vectors are used to predict the movement. f_n is the same, however instead of comparing directly with f_{n+1} the motion vectors are used to translate a location in f_n to the corresponding location in f_{n+1} before the difference is computed.

3.2.3 Evaluation of NLF identification methods

Evaluation of the NLF identification methods is necessary to identify the best method and its accuracy. The evaluation was done with a benchmark which compares the real noise level of a video to its computed noise level function. The first step of the benchmark is to set up a few test videos, with known noise. This is achieved by adding a fixed amount of noise to a noise free video. The two computer-generated videos *ChinaSpeed* and *SlideEditing* were used, because of absence of noise in these two videos. Seven different versions with noise were generated for each of the videos, using the algorithm described in Section 3.4. The first version had a noise level one, the second video had a noise level two and so on. The next step is to compute the NLF for each video using one of the NLF identification methods. Then the NLF is compared to the actual amount of noise by computing the mean difference between the estimated NLF and the known noise level. The temporal NLF achieved a result closer to the real noise level and will therefore be the NLF estimating method to use, see section 4.2 for the detailed evaluation results. As described in Section 3.3.3 the videos were in an 8 bit quality and thus support a luminance range of 0-255, however the actual used range for the NLF in this project was 0-64 because a NLF range of 0-255 ended in too large variation. The 64 range was enforced by binning, i.e. grouping four consecutive luminance levels into a single level i.e., all pixels with luminance level 1,2,3 and 4 are assigned luminance level 1, all pixels with luminance level 5,6,7 and 8 are translated to have luminance level 2 and so on.

3.3 Denoising

Denoising is the second part of the algorithm and focuses on removing the noise from the video while preserving the video information. There are multiple types of denoising algorithms as discussed in section 2.4, therefore the first part is to identify the best denoising algorithm based on a few criteria:

- The amount of noise removed in the denoising process.
- The amount of video information preserved.
- The impact on the bit-rate which the denoising process has on the video.

3.3.1 Benchmark denoising algorithms

Without a proper denoising algorithm the whole process is destined to fail, therefore were two benchmarks constructed to test the quality of different denoising algorithms.

Synthetic benchmark

The first benchmark is a synthetic benchmark and it operates by adding artificial noise and then measuring the denoising tool's efficiency in removing it. The procedure of the benchmark is visualized in Figure 3.2. The first step of the benchmark is to set up a noise free video, called video $V_{original}$. $V_{original}$ is a computer-generated video and therefore does not have any of the natural image sensor noise. Then next step is to add noise to $V_{original}$, an example frame can be observed at Figure 3.3. Thereafter the denoising is applied and the image quality is then measured. PSNR and SSIM were the metrics used to measure the video quality difference between $V_{original}$ and the denoised video. The noise free videos used were the two computer generated videos *ChinaSpeed* and *SlideEditing* from the test suits described in Section 3.1.1. For each of the two computer generated videos two different NLF were added. The two NLF were extracted from other real videos to make sure the NLF matches a real NLF. The videos use to identify the NLF were chosen at random from the test suit. The two videos were *BQTerrace* and *BasketballDrill*.

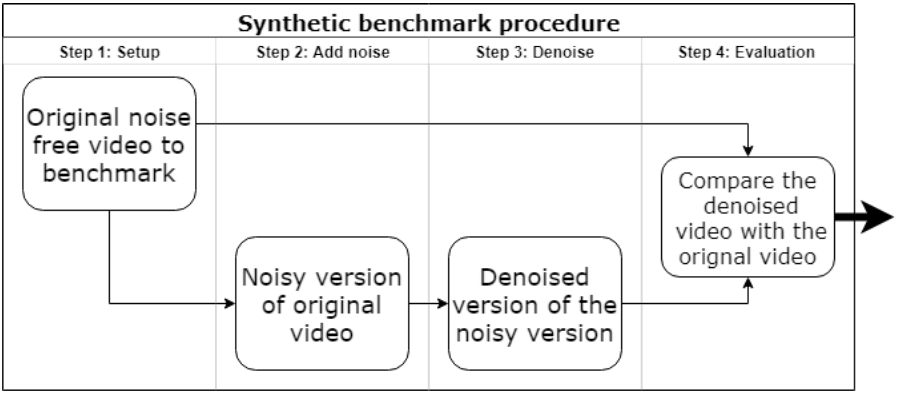

Figure 3.2. The procedure of the synthetic benchmark of the considered denoising methods.

Figure 3.3. A frame from ChinaSpeed with and without noise. The image to the left is the original frame of a noise free video and the right image is the same frame with added noise equivalent to the noise of *BQTerrace*.

Real data benchmark

The goal of the second benchmark, the real data benchmark, is to test both video quality and the amount of data which is needed to store the video information. The procedure of the benchmark is visualized in Figure 3.2. The approach of the benchmark is to encode two different versions of one video, call the original video $V_{original}$ and then compare the results of the encoded versions to the noisy original. The first encoded video is generated by encoding $V_{original}$ directly. The second video to be encoded is denoised before it's encoded. The denoised video is referred to as $V_{denoised}$. The hypothesis is that the encoded version of $V_{denoised}$ will have a lower BD-rate compared to the encoded version of $V_{original}$ due to the encoder's poor handling of noisy data. Then $V_{original}$ and $V_{denoised}$ were encoded using the codec JEM version 4.1, call the encoded versions $VE_{original}$ and $VE_{filtered}$. The encoding was done using the four different video qualities settings QP22, QP27, QP32 and QP37, where QP22 results in a high video quality encoding and QP37 in a low quality [WK08]. Finally, all the versions of $VE_{original}$ and $VE_{filtered}$ are compared to $V_{original}$ using the metric BD-rate. To save time all videos in the data benchmark were cropped to a video size of 512x384, nevertheless the final best result was validated in the original video resolution.

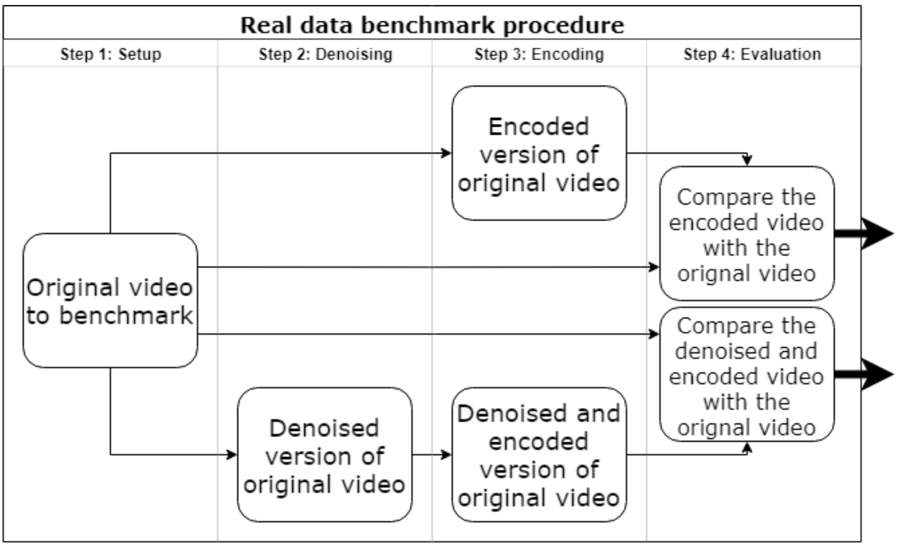

Figure 3.4. The procedure of the real data benchmark of the considered denoising methods.

3.3.2 Denoising algorithms

Both benchmarks evaluated a few different denoising algorithms namely, i) *Owdenoise* a denoising algorithm using wavelet transform, ii) *Hqdn3d* a denoising algorithm focusing on both the spatial and temporal domain, and lastly iii) *MCSpudsmod*, a temporal denoising algorithm with motion compensation. These algorithms are part of different encoding tools, so to use these algorithms the following tools were used: *FFmpeg* a multimedia framework which contains *Owdenoise* and *Hqdn3d* [Bel16] and the tool *AviSynth* a tool for video post-production which contains the denosing tool *MCSpudsmod* [RG03]. To each optimal results of the denoising tools different parameter of the tools were benchmarked to detect not only the best denoising tool but also the best setting for each tool. The following section describes the denoising algorithms and how their parameters were optimized.

Owdenoise

Owdenoise is a denoising algorithm using the wavelet transform to reduce noise while keeping most of the information of the video. *Owdenoise* has three different parameters controlling the denoising: first *Depth* which controls how much noise can be removed from low frequency components, secondly *Luma_strength* which controls how much brightness of the video can be altered during denoising, and the last is *Chroma_strength* which controls how much the color information of the video can be altered during denoising [FFm17]. *Owdenoise* was optimized by first finding an optimal *Luma_strength* level, then different *Depth* values were tested in combination with the best *Luma_strength* in order to improve the result.

Table 3.1. *Owdenoise parameters table*

Name	**Description**	**Range**
Depth	Larger depth values will denoise lower frequency components more, but increase the computational intensity.	8-16
Luma_strength	Specifies how much brightness of the video can be altered in the denoising, the higher value the more brightness information will be altered during denoising.	0-1000
Chroma_strength	Specifies how much color information of the video can be altered in the denoising, the higher value the more color information will be altered during denoising	0-1000

High Quality 3D Denoiser

High Quality 3D Denoiser (Hqdn3d) is a high precision 3D denoise algorithm operating in both the spatial and temporal domain. *Hqdn3d* uses nonlinear filtering to denoise similar to the procedure described in section 2.4.1. The strength of the denoiser is controlled by four parameters, two temporal and two spatial, see Table 3.2. *Hqdn3d* was optimized by first finding the optimal spatial and temporal parameters separately and then trying to find an optimal combination for those two. For more info about *Hqdn3d* see [Bel16].

Table 3.2. *Hqdn3d parameters table*

Name	Description	Range
Luma_spatial	Specifies the spatial denoising strength for the brightness of the video. The higher the value the more the brightness can be altered during denoising.	0-255
Chroma_spatial	Specifies the spatial denoising strength for the color information of the video. The higher the value the more the color information can be altered during denoising.	0-255
Luma_tmp	Specifies the temporal denoising strength for the brightness of the video. The higher the value the more the brightness can be altered during denoising.	0-255
Chroma_tmp	Specifies the temporal denoising strength for the color information of the video. The higher the value the more the color information can be altered during denoising.	0-255

MCSpudsmod

MCSpudsmod is a motion compensated denoising tool, focusing on denoising effectiveness at the cost of speed. *MCSpudsmod* is a merge of many different tools, the most prominent being the denoising script *Mvdegrain*, a nonlinear temporal denoising tool which uses motion vectors for increased accuracy in the denoising. *Mvdegrain* operates by computing a weighted mean over multiple frames, where the weight scales with the similarity of the denoised pixel. *Mvdegrain* also uses a wide set of thresholds to detect scene changes and movement in the video. *MCSpudsmod* supports a wide array of parameters to change different settings. The setting used in this project can be seen in Table 3.3. All the setting for *MCSpudsmod* can be found in [Spu16]. The parameter *sharpp* controlling the sharpening is by default turned on in *MCSpudsmod*, however it was disabled in all runs.

Table 3.3. *MCSpudsmod parameters table*

Name	Description	Range
Strength	Sets the default values for all other parameters, the higher Strength value the more the video will be altered during denoising.	0-6
Frames	Frames sets the amount of forward and backward frames which will be analyzed when denoising, a value of 2 indicates that the two previous and the 2 following frames will be used. A Frame setting of 4 means a combination of setting 2 and 3 will be used.	1-4
Thsad	Threshold which controls the weights of the non-linear filter. A high Thsad value will allow the data to be altered more compared to a low Thsad value.	0-1000

3.3.3 10bit videos

The tool *AviSynth*, which was used to run the denoising tool *MCSpudsmod* uses a script called *RawSource* to open raw videos, however *RawSource* does not support 10bit color range video as of version 26 [Chi17]. Therefore, the 10bit videos were converted to 8bit video before being denoised by *MC-Spudsmod*. The conversion was done utilizing the tool *FFmpeg*. Then is the video converted back into 10bit color range before being decoded. Because of the conversion 10bit to 8bit intensity range, some data is lost. 8bit range is [0,255] and 10bit has a range of [0,1023] which is four times higher resolution, so only every forth value is represented. This mens that after a conversion back and forth the intensity values can be the color value be off 2 points on the 1024 scale.

3.3.4 Denoising tool used and its settings

MCSpudsmod achieved the best PSNR and SSIM score in the synthetic benchmark and the lowest BD-rate in the real data benchmark and was therefore the denoising tools used in the rest of this project. The best setting varied slightly for the different benchmarks where the best parameter settings for the synthetic benchmark was Thsad=400 and Frame=3, for the data benchmark it was Thsad=300 and Frame=4. Of all the combinations Frame=4 and Thsad=300 achieved the best performance on the combination, of the two benchmarks and is therefore the setting used, see Section 4.3 for detailed presentation of the results.

3.4 Reapplying noise

The final part of the algorithm is to reapply the noise, this was done with the information from the NLF. Gaussian noise describes the noise of digital cameras as seen in Section 2.3.6 and is therefore the noise type used when reapplying noise. The following pseudo code describes how the noise was added for each pixel in a frame using the NLF.

```
procedure NOISE APPLIER
    NLF ← The noise level function of the frame
    frame ← the frame
    width ← the width of the frame
    height ← the height of the frame
    for w = 1 : width do
        for h = 1 : height do
            luminance_value ← get_luminance_value(frame[w][h])
            noise_level ← NLF(luminance_value)
            frame[w][h] ← frame[w][h] +
normal_random_value(-1,1) ∗ noise_level
```

Algorithm 1: adding noise to a frame using the NLF

The exact NLF used in this project was the NLF identified on the second frame for each video. The motivation to use one NLF in this project while several are available will be discussed in the Section 5.

3.4.1 Limit to the amount of noise added

Extracting an accurate NLF is necessary to generate accurate noise, however it's non-trivial to do so and all presented NLF identifying methods have some limitations. The spatial method is dependent on finding homogeneous regions, however the computed NLF becomes inaccurate if there are no homogeneous regions or if the algorithm fails to identify the regions. The temporal NLF is dependent on accurate motion vector prediction, which can be hard to estimate when there is a lot of noise; furthermore the method is inapplicable when there is a new scene in the video. Because of these limitations the NLF values can be too large resulting in too much added noise. One method to tackle this problem is to limit the maximum value of the NLF. The method is based on the difference between the original video and the encoded filtered video. The difference between these two videos comes from two different sources, one being data loss due to the denoising and the other being data loss due to encoding. If the amount of added noise is more than the difference of the original and the encoded video then too much noise is added. Therefore, using the differences between the two videos a hard limit of the amount of noise added to the video can be set. The difference is computed by comparing frame by frame. The computed difference was stored in the same format as a NLF, meaning the mean difference of the two videos was computed for every intensity level. By this, every value of the NLF can be limited per intensity level. This method has an additional advantage over the pure NLF, it's dependent on the amount of noise removed by the denoiser, measuring the maximum actual noise removed rather than the total amount of noise. If the denoiser did a poor job, not all noise was removed then the full amount of noise should not be added back in the renoising phase.

3.4.2 Evaluating the final video

After the noise is added all steps are completed. The final result was evaluated through a subjective survey. The survey was conducted in an Ericsson laboratory on a 4k TV the 21-06-2017. The survey consisted of 6 participants of different ages. The video *BQTerrace* and two additional videos chosen at random from the video set described in Section 3.1.1 were used in the survey. The videos used were *BQTerrace*, *Cactus* and *CampfireParty*; these three videos will be referred to as the original videos. For each of the original videos two different sets of encoded versions were used, one where the full algorithm described in this thesis was used and one where the videos were encoded and decoded without any denosing and renosing. For each set four different quality settings were used, resulting in a group of nine versions for each video including the original version. The different quality setting used were controlled by Quantization Parameter (QP), the QPs used were: QP22, QP27, QP32 and QP37. Before each video was displayed the original was displayed as a reference point. The order in which the videos were displayed was random, the final order is presented in Appendix A. *The survey participant answered the question: How close is the video to the original.* Each video was ranked from 1-5 where 1 is the lowest possible score and 5 the highest.

4. Results

4.1 Overview of main result

This section presents the result of the thesis. The following three Figures 4.1, 4.2 and 4.3 display the result of the survey in relation to the bit-rate of each video. From the figures a constant subjective score improvement for the re-noised technique cam be observed. Furthermore, a higher subjective score is observed at almost every bit-rate level.

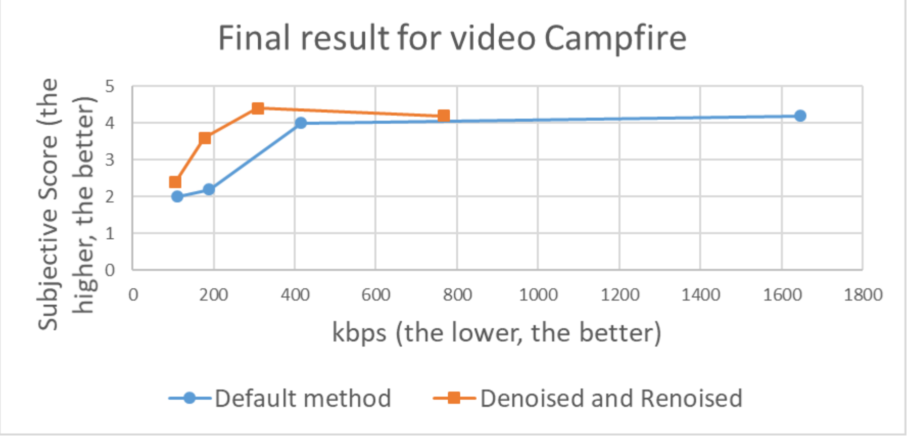

Figure 4.1. Final result for CampfireParty. The result of the subjective survey in relation to the bit-rate for the video CampfireParty

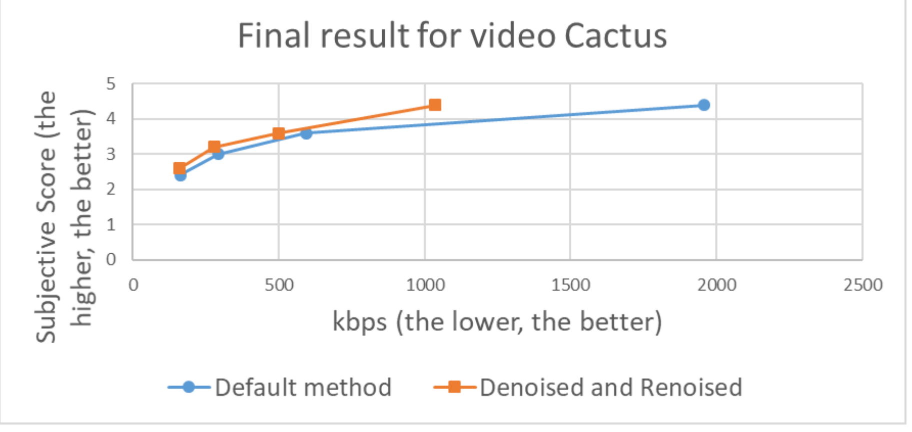

Figure 4.2. Final result for Cactus. The result of the subjective survey in relation to the bit-rate for the video Cactus.

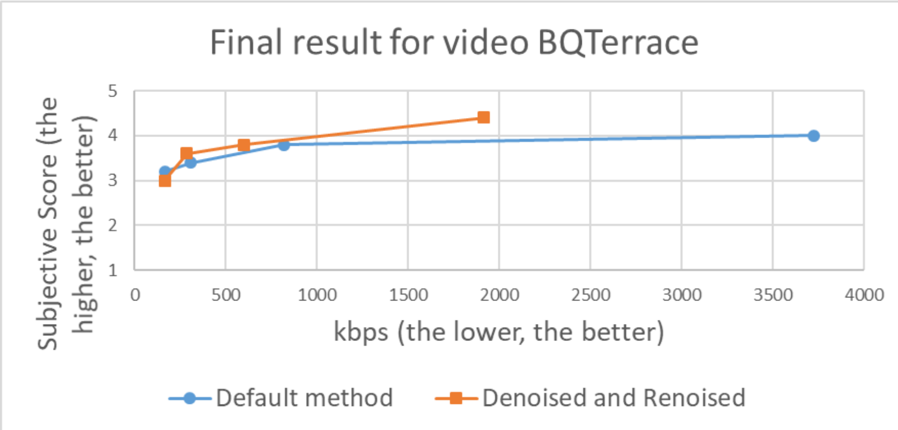

Figure 4.3. Final result for BQTerrace. The result of the subjective survey in relation to the bit-rate for the video BQTerrace.

4.2 Noise level function identification

The result of the two NLF identifications method, i.e. spatial and temporal NLF identification and their respective absolute error are presented in figure 4.4. The top two figures display the mean noise level identification for different noise levels. These images show a linear increase in identified noise level compared to real noise level for both the temporal and spatial method. The spatial method identifies more noise for low levels of noise compared to the temporal algorithm, however the temporal method finds more noise at high noise levels compared to the spatial method. The two bottom figures display the absolute error of the NLF compared to the real noise level. The absolute error grew with the noise level apart from a few exceptions. On average the noise level identified by the spatial method was of 0.85 noise levels and the temporal method was off by 0.62 noise levels. For exact values of the NLF please see appendix D.

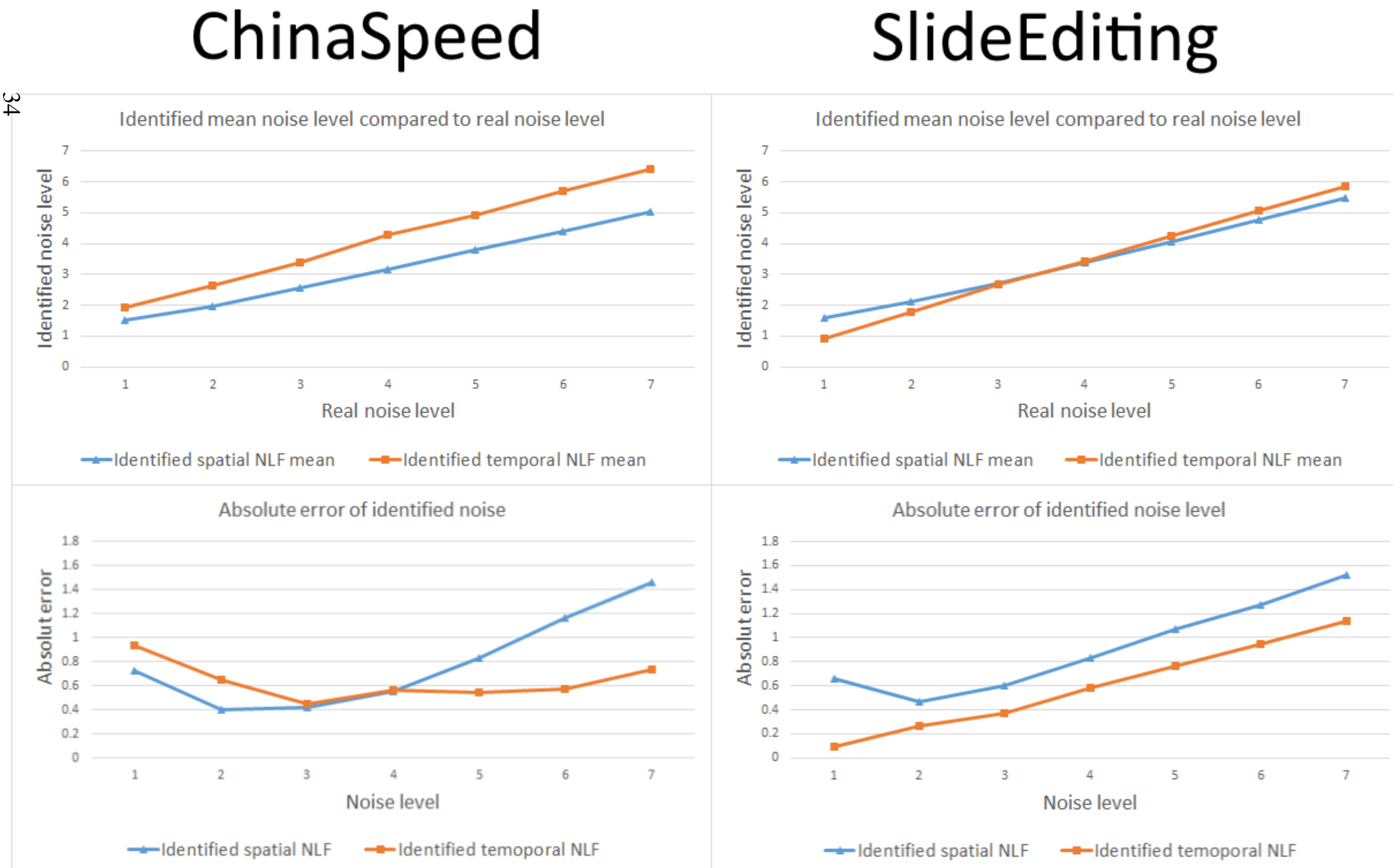

Figure 4.4. The result of the noise level function evaluation. The two top images display the mean noise level of the noise level function compared to the real noise level. The two bottom images display the absolute error between the noise level function and the real noise level.

4.3 Denosing

4.3.1 Synthetic benchmark

The PSNR and SSIM score of the synthetic benchmark are displayed in Figures 4.6 and 4.7. *MCSpudsmod* achieved the highest score in both metrics. *MCSpudsmod* achieved a mean PSNR score of 42.1 and SSIM score of 0.972 compared to a PSNR score of 37.2 and SSIM score of 0.919 if no denoising was used. *HQDN3D* achieved a mean PSNR score of 40.2 and an SSIM score of 0.963 and *Owdenoise* achieved a mean PSNR score of 39.1 and an SSIM score of 0.957, respectively, thus all denoising tools improved the scores compared to no denoising. The best setting for *MCSpudsmod* in the Synthetic benchmark was the following:

- Frame: 3
- Strength: 1
- Thsad: 400

An example frame of the Synthetic benchmark can be seen in Figure 4.5. The results of every benchmarked setting are found in Appendix B.

Figure 4.5. Results from the synthetic benchmark. The image to the upper left is the original frame of a noise free video, the image to upper right is the same frame with added noise equivalent to a noise level of 5. The image to the lower left is the denoised version of the right upper image using *MCSpudsmod*.

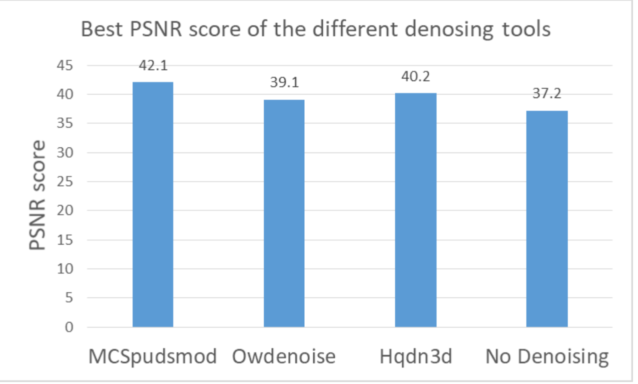

Figure 4.6. The mean PSNR score of both benchmarked videos for each denoising tool's optimal parameter setting in the synthetic benchmark.

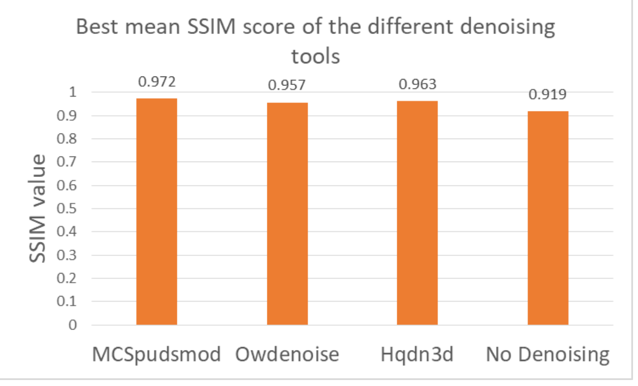

Figure 4.7. The mean SSIM score of both benchmarked videos for each denoising tool's optimal parameter setting in the synthetic benchmark.

4.3.2 Real data benchmark

The second denoising benchmark measuring the BD-rates, showed similar results as the synthetic benchmark where *MCSpudsmod* preformed the best followed by *HQDN3D* and *Owdenoise*. In figure 4.8 the result can be observed, where the best *MCSpudsmod* setting resulted in a BD-rate of -2.7% whereas the best result for *HQDN3D* was -0.005% BD-rate and *Owdenoise* had a BD-rate of 6.9%. The best setting for *MCSpudsmod* was similar to the ones of the synthetic benchmark except for the *Frame* setting which had a best value of 4 instead of 3 and Thsad 300 instead of 400. The best setting for *MCSpudsmod* was the following:

- Frame: 4
- Strength: 1
- Thsad: 300

The results for each individual video for *MCSpudsmod* are given in table 4.1. The best BD-rate of -11.2% was archived by *Cactus*, the worst score was for *Tango* with an BD-rate of 1.5%. The full result of the benchmark can be observed in Appendix C.

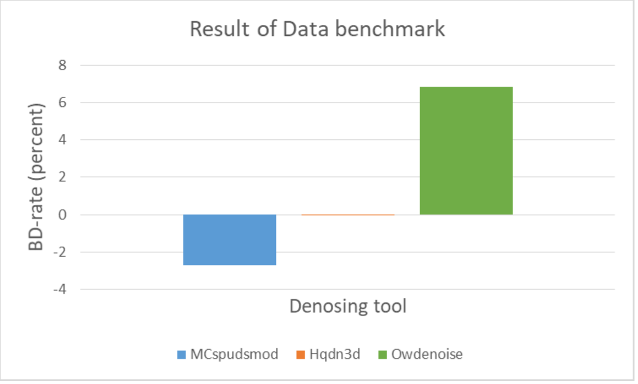

Figure 4.8. Best achieved BD-rate for the different denoising algorithms and different parameter settings in the data benchmark.

4.3.3 Best denosing tool

Frame=4 and Thsad=300 showed to be the best performance combination of all the combinations of *MCSpudsmod* parameter settings in the two benchmarks. The BD-rate was -2.7% in the real data benchmark and a PSNR score of 40.9 and SSIM score of 0.962 was achieved in the synthetic benchmark. All results of the benchmarks can be observed in Appendix B and C.

Table 4.1. *Results of MCSpudsmod in the data benchmark.*

Video name	Resolution	Frame rate	Bit-rate	BD-rate
Tango	4096x2160	60	10	1.429
ToddlerFountain	4096x2160	60	10	1.356
CampfireParty	3840x2160	30	10	-0.137
Drums	3840x2160	100	10	-7.525
CatRobot	3840x2160	60	10	-6.021
DaylightRoad	3840x2160	60	10	-10.863
TrafficFlow	3840x2160	30	10	-4.514
Kimono	1920x1080	24	8	-1.729
ParkScene	1920x1080	24	8	-4.657
Cactus	1920x1080	50	8	-11.229
BQTerrace	1920x1080	60	8	-10.85
BasketballDrive	1920x1080	50	8	-0.394
FourPeople	1280x720	60	8	-4.825
Johnny	1280x720	60	8	-7.088
KristenAndSara	1280x720	60	8	-5.651
BQMall	832x480	60	8	-1.088
PartyScene	832x480	50	8	1.885
RaceHorses	832x480	30	8	-0.462
BasketballDrill	832x480	50	8	0.587
BasketballDrillText	832x480	50	8	0.642
BQSquare	416x240	60	8	3.346
RaceHorses	416x240	30	8	-0.107
BasketballPass	416x240	50	8	1.15
BlowingBubbles	416x243	50	8	0.061

4.4 Reapplying noise

The last part of the algorithm was to reapply noise. An example can be observed in Figure 4.9 where a frame with the added noise is shown.

Figure 4.9. Noise reapplied. The left image is a denoised and encoded version of BQTerrace, the right image is the same image with added noise. Please note that the noise is most visible on the pillar on the left side of the images.

The final state of the project was evaluated with a subjective survey. The result of the survey can be observed in Figure 4.10 and 4.11. The denoising-renosing method achieved a consistently better score. The subjective score scaled with the QP for both methods. In total the BD-score for *BQSquare* was -31.76%, -56.62% for *CampfireParty* and -24.61% for *Cactus*. For the detailed evaluation results please see Appendix A.

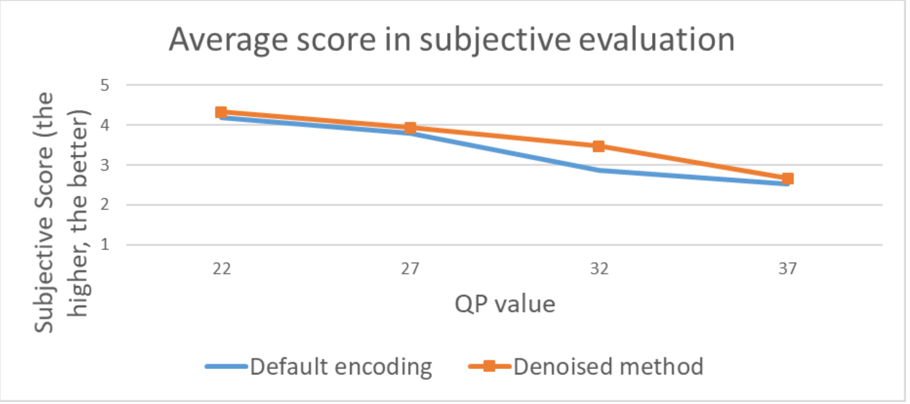

Figure 4.10. Results of subjective survey, the subjective score for each of the different QPs, ranging from 1-5.

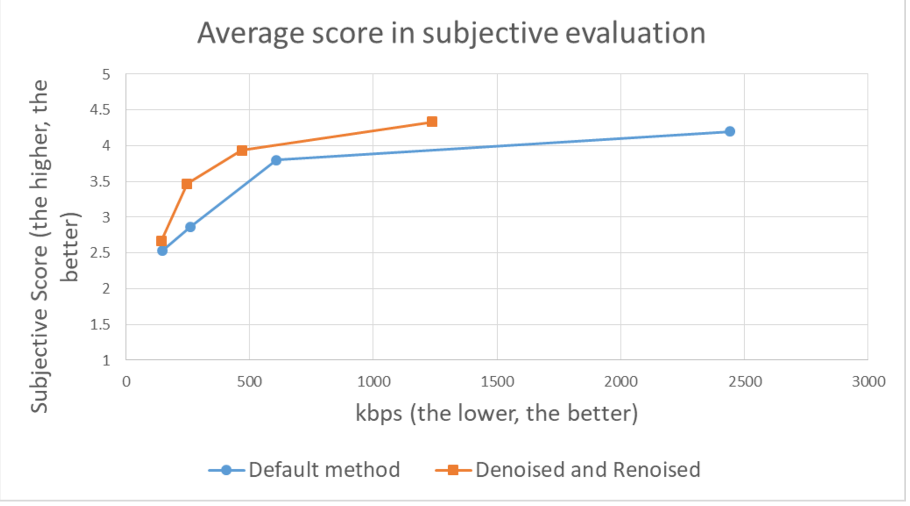

Figure 4.11. Results of subjective survey, the mean subjective score and the corresponding bit-rate for all videos.

5. Discussion

The denoising-renoising tool developed in this MSc project shows to be successful in accomplishing all of the project aims; the bit-rate is reduced and the image quality is improved. Evaluation of the combined visual improvement and the reduced bit-rate showed that the achieved BD-rate improvement ranges from -25% to -56%, i.e, the same visual quality can be achieved with up to 56% lower bit-rate. The gained image quality is observed for all video settings, while the reduction in bit-rate increased with the decreasing QPs. The encoding of high QPs reduces noise to a minimum. This explains why improvement in bit-rate by the denosing was low for the videos encoded with a poor video quantile and high for videos with high QP. This result is similar to *Oh et al. 2009* [OLK09] where a similar method was used focusing on noise from analog cameras rather than digital cameras. *Oh et al. 2009* achieved a bit-rate saving of 35% for high quality settings (QP 20) compared to a bit-rate saving of 22% for medium quality settings (QP 28). The bit-rate saving of this project was 49% for high quality settings (QP 22) and 18% for medium quality settings (QP 27).

5.1 Noise Level function identification

Overall the NLF identification methods were successful, the error of the temporal method was 0.62 noise levels on average and the spatial 0.82 noise levels on average. The two methods used to identify the NLF had their strengths and weaknesses. The tool for detecting spatial NLF depends on accurately identifying homogeneous areas. However, this tool is based on a crude assumption that 10% of the area is homogeneous. A more accurate spatial NLF identification algorithm was presented in [SAD16], were the uniformity of the pixels in the region was checked to identify the homogeneous areas. The presented temporal NLF identification method requires accurate motion vector prediction. The more noise, the harder it is to accurately predict the motion vector which decreases the accuracy of the NLF detection.

5.2 Denosing

The result of the synthetic benchmark was positive, i.e. the quality of the video can be objectivity improved by utilizing the denoising tools. *MCSpudsmod* achieved the best score in the synthetic benchmark followed by *HQDN3D* and then *Owdenoise*. The optimal parameter setting was slightly different for the two metrics, PSNR and SSIM, nevertheless the settings which had the best PSNR score had an SSIM score of 0.972 compared to the best SSIM score of 0.974, a negligible difference.

MCSpudsmod also achieved the best BD-rate in the data benchmark, with a mean BD-rate of -2.7%. The result proves the hypothesis, that denoising can be used to improve the bit-rate for a given video quality in a modern encoder. However, the results were not consistent, some videos had an improved BD-rate and some a decreased BD-rate. The BD-rate was better for high resolution videos in general. For example, videos with a resolution of 1280x720 or higher had an average BD-rate of -4.84% compared to a BD-rate of 0.66% for videos with lower resolutions than 1280x720. One possible reason for the difference is that for low resolution videos there are fewer pixels for every object in the video making it harder to distinguish noise from information and thus decreasing the accuracy of the denoising tool. The results differed within resolution groups, for example in group B (videos with resolution of 1920x1080) where *Cactus* had a BD-rate of -11.2% while *BasketballDrive* only had -0.4% and in group A (videos with resolution of 3840 x 1600) were *DaylightRoad* had an BD-rate of -10.8% compared to *Tangos* 1.4%. One possible explanation is the different amount of movement in the video; in *Cactus* there is very little movement compared to *BasketballDrive* which has a lot of movement. *MCSpudsmod* relies on temporal data to predict the noise, more movement makes it harder to distinguish noise from information resulting in worse denoising. Of the 24 videos there are 7 with the BD rate which decreased due to denoising, which indicates that denoising is not always a suitable choice, if the goal is a better video quality bit-rate ratio.

The tuned denoising tools settings from the two different denoising benchmarks were similar, but not exactly the same. The optimal Thsad threshold was higher for the synthetic benchmark, meaning more data can be removed during the denoising process. One explanation for the difference is that in the real data benchmark is the denosing combined with an encoding process, which further reduces the amount of noise. The combination of a denoising filter with a high Thsad threshold and encoding might result in too much information loss during denoising. A second reason for the difference is how the video quality is computed. In the real data benchmark video quality of a denoised and encoded video is computed by comparing it to the original video. The original video contains noise so removing noise results in a lower video quality score.

5.2.1 Noise level function overhead and usages

The NLF uses 64 different luminance levels in this project, each value requires 1 byte, and so the total overhead of a NLF is 64 bytes. Only one NLF was used in this project so the total overhead of the methods was 64 bytes which is negligible. The used NLF was the one from the second frame in the video, however if the second frames do not have the full luminance range, for example a black frame, the NLF will be poorly estimated. A possible improvement is to compute the NLF as an average over all frames to ensure that NLF covers all luminance values. Using only one NLF for the entire video is not always ideal e.g., when different cameras are used throughout the video. Using different cameras results in different CRF and therefore different NLF. One solution could be to use a new NLF for each frame. For the video *CampfireParty* which runs at 30 frames per second it would result in a 240 bytes extra overhead per second. For *CampfireParty* at QP 37 the overhead would be around 2%, whereas for QP 22 would it be 0.25%. The extra overhead does not change the overall result of the method, however it is worth investigating if more NLF frames increase the overall visual performance.

5.3 Subjective evaluation

The result of the subjective evaluation confirms on average improved video quality. The perceived video quality is improved from 3.35 to 3.6 average score on a subjective 1-5 scale. *CampfireParty* achieved the best score with an improved score from 3.1 to 4. These scores were achieved using the same QP value, however the bit-rate was also significantly reduced. There were some oddities in the evaluation, for *BQTerrace* where the original video scored 3.6 when compared to itself, instead of the expected score close to 5. This big difference could possibly invalidate the *BQTerrace* test set of the survey, nevertheless the conclusion is still the same with or without *BQTerrace* and thus it does not matter if *BQTerrace's* result is invalid or not. Another oddity was the loss of subjective video quality for *CampfireParty* between QP value 27 and 22, this difference is likely due to chance, however it's necessary to further investigate this oddity if this error is persistent in future investigations. Preferably more videos should be subjectively tested to statistically secure the evaluation, yet the result of the evaluation is still strong enough to indicate a positive result. The video quality was improved using the denoising and renosing technique.

6. Conclusion

In conclusion, the aims of this project were successfully achieved; with an algorithm which first removed noise and then reapplied it after decoding, the video quality could be improved and at the same time a lower bit-rate can be used. The denoising tool *MCSpudsmod* removed a significant amount noise while preserving the image information. A temporal NLF identification method was developed to estimate the NLF. Out of the two tested methods to estimate NLF, a spatial and a temporal one, the temporal method showed to be more accurate and is included in the final algorithm. The estimated NLF can then be used to reapply noise in a way visibly pleasant for the viewer. Furthermore, it was shown that the bit-rate at a fixed quality level (BD-rate) is improved using denoising.

Acknowledgements

I would like to thank Per Wennersten, my supervisor at Ericsson Research for all the support and valuable discussions.

Furthermore, would I like to thank my reviewer Natasa Sladoje at the Centre for Image Analysis, Dept. for Information Technology, Uppsala University for her help and support in making of this thesis.

I also would like to thank Annemieke Gärdenäs for her support in writing of this thesis.
Finally, my thanks also go to my family and friends for all the help and support.

Tack så mycket!
Anders

References

[Bar13] Tudor Barbu. Variational image denoising approach with diffusion porous media flow. In *Abstract and Applied Analysis*, volume 2013. Hindawi Publishing Corporation, 2013.

[Bel16] Fabrice Bellard. FFmpeg filters documentation. `https://ffmpeg.org/ffmpeg-filters.html#hqdn3d-1`, 2016. [accessed 28-Aug-2017].

[Bjo01] Gisle Bjontegaard. Calcuation of average PSNR differences between rd-curves. *Doc. VCEG-M33 ITU-T Q6/16, Austin, TX, USA, 2-4 April 2001*, 2001.

[BKE+95] James C. Brailean, Richard P. Kleihorst, Serafim Efstratiadis, Aggelos K. Katsaggelos, and Reginald L. Lagendijk. Noise reduction filters for dynamic image sequences: A review. *Proceedings of the IEEE*, 83:1272–1292, 1995.

[Bro13] Blain Brown. *Cinematography: theory and practice: image making for cinematographers and directors*. Crc Press, 2013.

[Buc70] Richard S. Bucy. Linear and nonlinear filtering. *Proceedings of the IEEE*, 58(6):854–864, 1970.

[CB13] Miguel Colom and Antoni Buades. Analysis and extension of the percentile method, estimating a noise curve from a single image. *Image Processing On Line*, 3:332–359, 2013.

[CEPY05] Tony Chan, Selim Esedoglu, Frederick Park, and A Yip. Recent developments in total variation image restoration. *Mathematical Models of Computer Vision*, 17(2), 2005.

[Chi17] Chikuzen. Source filter plugin for loading raw video data from files. `http://avisynth.nl/index.php/RawSource26`, 2017. [accessed 14-Aug-2017].

[CLYY12] Xiaogang Chen, Feng Li, Jie Yang, and Jingyi Yu. A theoretical analysis of camera response functions in image deblurring. In *European Conference on Computer Vision*, pages 333–346. Springer, 2012.

[CPW11] G Cermak, M Pinson, and Stephen Wolf. The relationship among video quality, screen resolution, and bit rate. *IEEE Transactions on Broadcasting*, 57(2):258–262, 2011.

[Eri17] Ericsson. Ericsson mediafirst video processing. `https://www.ericsson.com/ourportfolio/products/mediafirst-video-processing?nav=productcategory007`, 2017. [accessed 13-Apr-2017].

[FFm17] FFmpeg. owdenoise. `https://peg.org/ffmpeg-filters.html`, 2017. [accessed 13-Aug-2017].

[For05] The DVD Forum. 29th steering committee meeting (february 23, 2005), 2005.

[HI17] Fraunhofer Heinrich and Hertz Institute. JVET JEM software. `https://jvet.hhi.fraunhofer.de/`, 2017. [accessed 11-Aug-2017].

[HK94] Glenn E. Healey and Raghava Kondepudy. Radiometric CCD camera calibration and noise estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):267–276, 1994.

[HP12] B.G. Haskell and A. Puri. *Chapter 2 MPEG Video Compression Basics*. Springer, 2012.

[HZ10] Alain Hore and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2366–2369. IEEE, 2010.

[Jac11] Keith Jack. *Video demystified: a handbook for the digital engineer*. Elsevier, 2011.

[Kod01] Eastman Kodak. *CCD Image Sensor Noise Sources*, 2001.

[KOS10] Michihiro Kobayashi, Takahiro Okabe, and Yoichi Sato. Detecting forgery from static-scene video based on inconsistency in noise level functions. *IEEE Transactions on Information Forensics and Security*, 5(4):883–892, 2010.

[LAG13] Joan Cooper Llach, Jeffrey Allen, and Cristina Gomila. Film grain simulation for normal play and trick mode play for video playback systems, May 21 2013. US Patent 8,447,124.

[LFSK06] Ce Liu, William T Freeman, Richard Szeliski, and Sing Bing Kang. Noise estimation from a single image. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 901–908. IEEE, 2006.

[LGLS08] Peter M. Levine, Ping Gong, Rastislav Levicky, and Kenneth L Shepard. Active CMOS sensor array for electrochemical biomolecular detection. *IEEE Journal of Solid-State Circuits*, 43(8):1859–1871, 2008.

[NG96] Mark Nelson and Jean-Loup Gailly. *The data compression book*, volume 2. M&t Books New York, 1996.

[OLK09] Byung Tae Oh, Shaw-min Lei, and C-C Jay Kuo. Advanced film grain noise extraction and synthesis for high-definition video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(12):1717–1729, 2009.

[OS13] Jens-Rainer Ohm and Gary J. Sullivan. High efficiency video coding: the next frontier in video compression [standards in a nutshell]. *IEEE Signal Processing Magazine*, 30(1):152–158, 2013.

[RG03] B. Rudiak-Gould. Avisynth. `http://avisynth.nl/index.php/Main_Page`, 2003. [accessed 13-Apr-2017].

[Ric04] Iain E. Richardson. *H. 264 and MPEG-4 video compression: video* *coding for next-generation multimedia*. John Wiley & Sons, 2004.

[Ric17] Iain Richardson. H.264/AVC 4x4 transform and quantization. *Mathematical Models of Computer Vision*, 2017.

[SAD16] Camille Sutour, Jean-François Aujol, and Charles-Alban Deledalle. Automatic estimation of the noise level function for adaptive blind denoising. In *Signal Processing Conference (EUSIPCO), 2016 24th European*, pages 76–80. IEEE, 2016.

[SFD10] KR Spring, TJ Fellers, and MW Davidson. Introduction to charge-coupled devices. *MicroscopyU: the source for microscopy education Retrieved June*, 2:2010, 2010.

[SM99] J-L. Starck and Fionn Murtagh. Multiscale entropy filtering. *Signal Processing*, 76(2):147–165, 1999.

[Spu16] Didée Spuds. Mc spuds. `http://avisynth.nl/index.php/MC_Spuds`, 2016. [accessed 14-Aug-2017].

[WK08] Hanli Wang and Sam Kwong. Rate-distortion optimization of rate control for h. 264 with adaptive initial quantization parameter determination. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(1):140–144, 2008.

[WS98] David E. Wolf and Greenfield Sluder. *Video-Microscopy*. Springer Science & Business Media, 1998.

Appendices

Appendix A.
Subjective survey

Table 1. *Subjective survey, the table displays the order which the videos were displayed in the subjective survey. The number represent the QP quality, "denoised" indicates that the denosing and renosing technique was used.*

Bq	Cactus	Campfire
27denoised	37	27
32denoised	37denoised	32denoised
22	original	32
37	22	22denoised
original	22denoised	37denoised
27	32	22
32	27	37
37denoised	32denoised	27denoised
22denoised	27denoised	original

Table 2. *Result of subjective survey for participant 1, please see table 1 to see witch score translates to which video quality.*

Bq	**Cactus**	**Campfire**
5	4	5
5	3	5
4	5	4
5	5	5
3	4	4
5	4	5
4	5	3
5	4	5
4	3	5

Table 3. *Result of subjective survey for participant 2, please see table 1 to see witch score translates to which video quality.*

Bq	Cactus	Campfire
2	2	3
2	3	2
4	5	1
3	4	5
3	5	1
5	3	4
4	4	1
1	2	4
4	4	4

Table 4. *Result of subjective survey for participant 3, please see table 1 to see witch score translates to which video quality.*

Bq	Cactus	Campfire
4	2	4
3	2	4
4	5	3
3	5	4
5	5	2
3	3	5
4	3	3
3	3	4
5	3	5

Table 5. *Result of subjective survey for participant 4, please see table 1 to see witch score translates to which video quality.*

Bq	Cactus	Campfire
4	3	4
3	2	4
3	4	2
3	4	4
2	4	2
3	3	4
2	3	2
2	3	4
4	4	3

Table 6. *Result of subjective survey for participant 5, please see table 1 to see witch score translates to which video quality.*

Bq	Cactus	Campfire
4	1	4
5	3	3
5	5	1
2	4	3
5	4	3
3	2	3
3	3	1
4	4	5
5	4	5

Appendix B.
Result of synthetic benchmark

Table 7. *PSNR score in synthetic benchmark for video ChinaSpeed. hqdn3d has four settings, the two first control spatial and the two following control temporal. MCSpudsmod has three settings, the first controls the frame setting. The second is the strength setting and the third is Thsad. Owdenoise has two settings, Depth and Luma_strength.*

Denosing technique \ Noise source	BQTerrace	BasketballDrillText
No denosing	35.41	41
hqdn3d=00_22	36.008105	42.437365
hqdn3d=00_44	36.211605	42.56264
hqdn3d=00_88	36.717946	42.017055
hqdn3d=00_1212	37.007449	40.903983
hqdn3d=00_1616	37.011816	39.722874
hqdn3d=00_2020	36.796153	38.644587
hqdn3d=22_00	35.567997	41.698054
hqdn3d=44_00	36.483283	42.404663
hqdn3d=88_00	37.975172	40.226802
hqdn3d=1010_00	37.909809	38.869128
hqdn3d=44_1616	37.011816	39.722874
hqdn3d=88_1212	37.975172	40.226802
hqdn3d=88_1616	37.673419	39.102311
MCSpudsmod1_1	37.216042	42.902366
MCSpudsmod2_1	38.05812	43.329556
MCSpudsmod3_1	38.536745	43.451231
MCSpudsmod4_1	38.115657	43.75352
MCSpudsmod3_0	35.374904	41.350405
MCSpudsmod3_1	38.536745	43.451231
MCSpudsmod3_2	38.926856	40.537481
MCSpudsmod3_3	37.86782	38.870085
MCSpudsmod3_1_200	36.108488	43.493791
MCSpudsmod3_1_300	38.536745	43.451231
MCSpudsmod3_1_400	39.604561	42.508163
MCSpudsmod3_1_500	39.498819	41.567294
owdenoise=10_4	38.351287	41.332002
owdenoise=10_5	38.372735	40.281019
owdenoise=10_6	38.120507	39.284946
owdenoise=16_4	38.351287	41.332002
owdenoise=16_5	38.372735	40.281019
owdenoise=16_6	38.120507	39.284946
owdenoise=8_1	36.367797	42.341584
owdenoise=8_2	37.303146	42.790735
owdenoise=8_3	37.99186	42.291036
owdenoise=8_4	38.351277	41.332087
owdenoise=8_5	38.372803	40.28111
owdenoise=8_6	38.120587	39.285

Table 8. *PSNR score in synthetic benchmark for video Slideeddeting. hqdn3d has four settings, the two first control spatial and the two following control temporal. MCSpudsmod has three settings, the first controls the frame setting. The second is the strength setting and the third is Thsad. Owdenoise has two settings, Depth and Luma_strength.*

Denosing technique \ Noise source	BQTerrace	BasketballDrillText
No denosing	36.18	36.4
hqdn3d=00_22	37.08362	37.509457
hqdn3d=00_44	37.501097	37.974961
hqdn3d=00_88	38.602399	38.9797
hqdn3d=00_1212	39.58464	39.86168
hqdn3d=00_1616	40.324168	40.543756
hqdn3d=00_2020	40.83207	41.010853
hqdn3d=22_00	36.643197	36.973804
hqdn3d=44_00	38.048359	38.476832
hqdn3d=88_00	41.172361	41.33578
hqdn3d=1010_00	42.049317	42.206945
hqdn3d=44_1616	40.324168	40.543756
hqdn3d=88_1212	41.172361	41.33578
hqdn3d=88_1616	41.732399	41.86611
MCSpudsmod1_1	39.222348	39.129155
MCSpudsmod2_1	40.819029	40.392196
MCSpudsmod3_1	41.858431	41.176123
MCSpudsmod4_1	41.138427	40.683663
MCSpudsmod3_0	36.335914	36.510682
MCSpudsmod3_1	41.858431	41.176123
MCSpudsmod3_2	43.96448	43.997786
MCSpudsmod3_3	43.987161	44.207505
MCSpudsmod3_1_200	37.781337	38.470941
MCSpudsmod3_1_300	41.858431	41.176123
MCSpudsmod3_1_400	43.170534	43.202713
MCSpudsmod3_1_500	43.467184	43.644122
owdenoise=10_4	38.031507	38.117651
owdenoise=10_5	37.649919	37.683862
owdenoise=10_6	37.067076	37.075352
owdenoise=16_4	38.031507	38.117651
owdenoise=16_5	37.649919	37.683862
owdenoise=16_6	37.067076	37.075352
owdenoise=8_1	37.09407	37.328498
owdenoise=8_2	37.780775	37.999049
owdenoise=8_3	38.099123	38.257183
owdenoise=8_4	38.031471	38.11761
owdenoise=8_5	37.649907	37.683849
owdenoise=8_6	37.06707	37.075309

Table 9. *SSIM score in synthetic benchmark for video ChinaSpeed. hqdn3d has four settings, the two first control spatial and the two following control temporal. MCSpudsmod has three settings, the first controls the frame setting. The second is the strength setting and the third is Thsad. Owdenoise has two settings, Depth and Luma_strength.*

Noise / Denosing technique	BQTerrace noise	BasketballDrillText noise
No denosing	0.867	0.96
hqdn3d=00_22	0.8839	0.972574
hqdn3d=00_44	0.888186	0.973592
hqdn3d=00_88	0.898545	0.972366
hqdn3d=00_1212	0.906305	0.969008
hqdn3d=00_1616	0.910984	0.964758
hqdn3d=00_2020	0.913352	0.960149
hqdn3d=22_00	0.872525	0.966626
hqdn3d=44_00	0.89365	0.973404
hqdn3d=88_00	0.925155	0.962564
hqdn3d=1010_00	0.930272	0.953958
hqdn3d=44_1616	0.910984	0.964758
hqdn3d=88_1212	0.925155	0.962564
hqdn3d=88_1616	0.926339	0.958096
MCSpudsmod1_1	0.91023	0.977132
MCSpudsmod2_1	0.925987	0.980558
MCSpudsmod3_1	0.934155	0.981756
MCSpudsmod4_1	0.926922	0.982195
MCSpudsmod3_0	0.867747	0.963561
MCSpudsmod3_1	0.934155	0.981756
MCSpudsmod3_2	0.95324	0.97582
MCSpudsmod3_3	0.948677	0.96908
MCSpudsmod3_1_200	0.890453	0.980816
MCSpudsmod3_1_300	0.934155	0.981756
MCSpudsmod3_1_400	0.949351	0.979793
MCSpudsmod3_1_500	0.951725	0.977946
owdenoise=10_4	0.932878	0.969418
owdenoise=10_5	0.939437	0.964803
owdenoise=10_6	0.942867	0.959731
owdenoise=16_4	0.932878	0.969418
owdenoise=16_5	0.939437	0.964803
owdenoise=16_6	0.942867	0.959731
owdenoise=8_1	0.889691	0.970685
owdenoise=8_2	0.908122	0.973938
owdenoise=8_3	0.922639	0.972952
owdenoise=8_4	0.932945	0.969488
owdenoise=8_5	0.939522	0.964885
owdenoise=8_6	0.942956	0.959821

Table 10. *SSIM score in synthetic benchmark for video Slideeddeting. hqdn3d has four settings, the two first control spatial and the two following control temporal. MCSpudsmod has three settings, the first controls the frame setting. The second is the strength setting and the third is Thsad. Owdenoise has two settings, Depth and Luma_strength.*

Noise / Denosing technique	BQTerrace noise	BasketballDrillText noise
No denosing	0.919	0.931
hqdn3d=00_22	0.936248	0.952928
hqdn3d=00_44	0.940993	0.958113
hqdn3d=00_88	0.951854	0.965909
hqdn3d=00_1212	0.96029	0.971239
hqdn3d=00_1616	0.966181	0.974995
hqdn3d=00_2020	0.970289	0.977597
hqdn3d=22_00	0.927698	0.942302
hqdn3d=44_00	0.946605	0.962275
hqdn3d=88_00	0.977276	0.982199
hqdn3d=1010_00	0.98465	0.986784
hqdn3d=44_1616	0.966181	0.974995
hqdn3d=88_1212	0.977276	0.982199
hqdn3d=88_1616	0.980193	0.9841
MCSpudsmod1_1	0.953069	0.960792
MCSpudsmod2_1	0.965298	0.969363
MCSpudsmod3_1	0.971644	0.973609
MCSpudsmod4_1	0.967114	0.971095
MCSpudsmod3_0	0.923107	0.935269
MCSpudsmod3_1	0.971644	0.973609
MCSpudsmod3_2	0.983645	0.986324
MCSpudsmod3_3	0.984443	0.986861
MCSpudsmod3_1_200	0.939017	0.960079
MCSpudsmod3_1_300	0.971644	0.973609
MCSpudsmod3_1_400	0.978408	0.981439
MCSpudsmod3_1_500	0.97997	0.983204
owdenoise=10_4	0.97093	0.976618
owdenoise=10_5	0.97652	0.979662
owdenoise=10_6	0.979912	0.981235
owdenoise=16_4	0.97093	0.976618
owdenoise=16_5	0.97652	0.979662
owdenoise=16_6	0.979912	0.981235
owdenoise=8_1	0.937057	0.949345
owdenoise=8_2	0.951556	0.962722
owdenoise=8_3	0.962723	0.97136
owdenoise=8_4	0.97093	0.976618
owdenoise=8_5	0.976521	0.979662
owdenoise=8_6	0.979913	0.981234

Appendix C.
Real data Benchmark

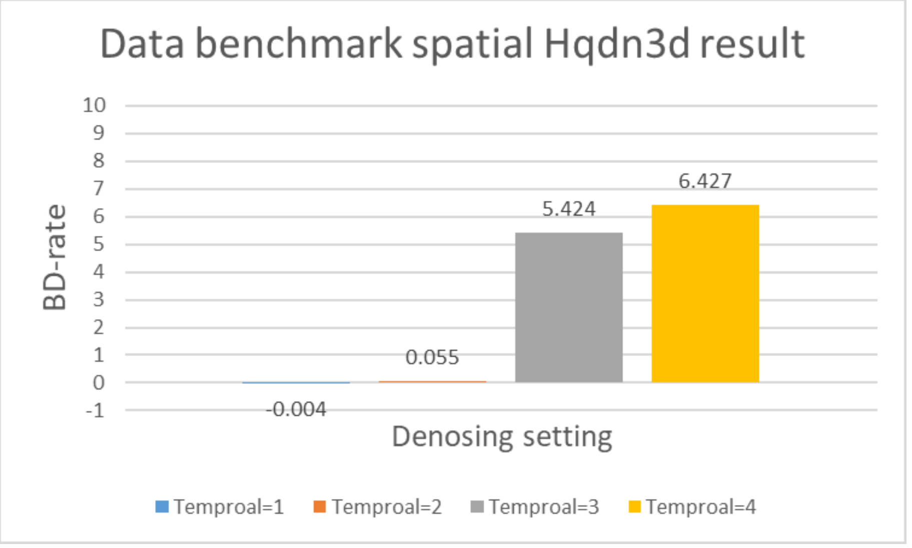

Figure 1. The result of the real data benchmark for different spatial settings for the denosing tool Hqdn3d

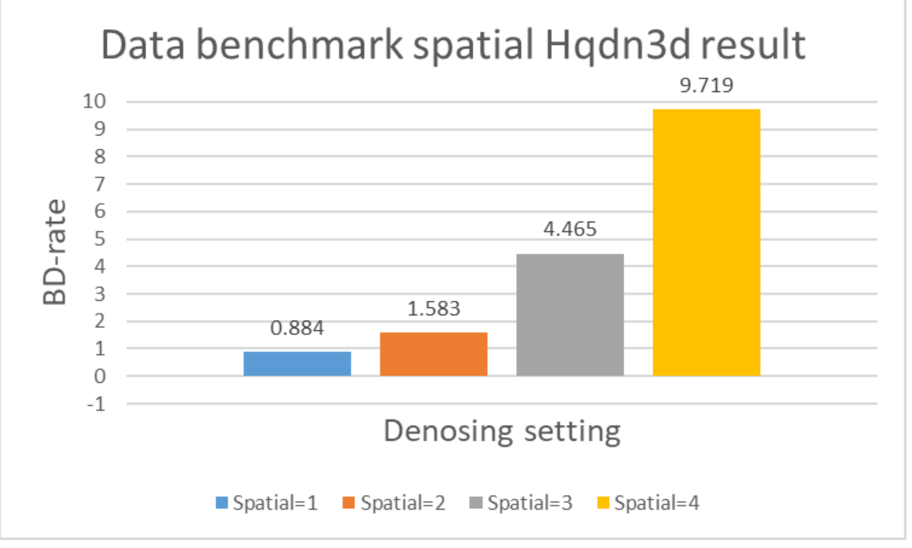

Figure 2. The result of the real data benchmark for different temporal settings for the denosing tool Hqdn3d.

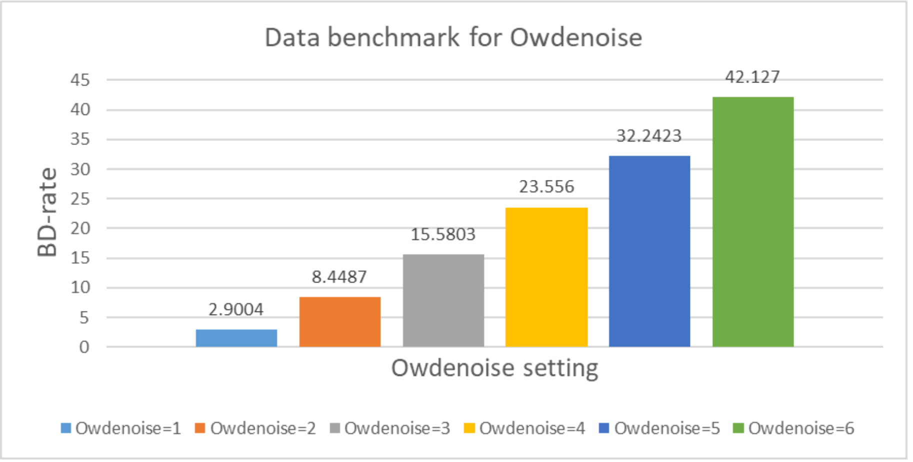

Figure 3. The result of the real data benchmark for different strength settings for Owdenoise.

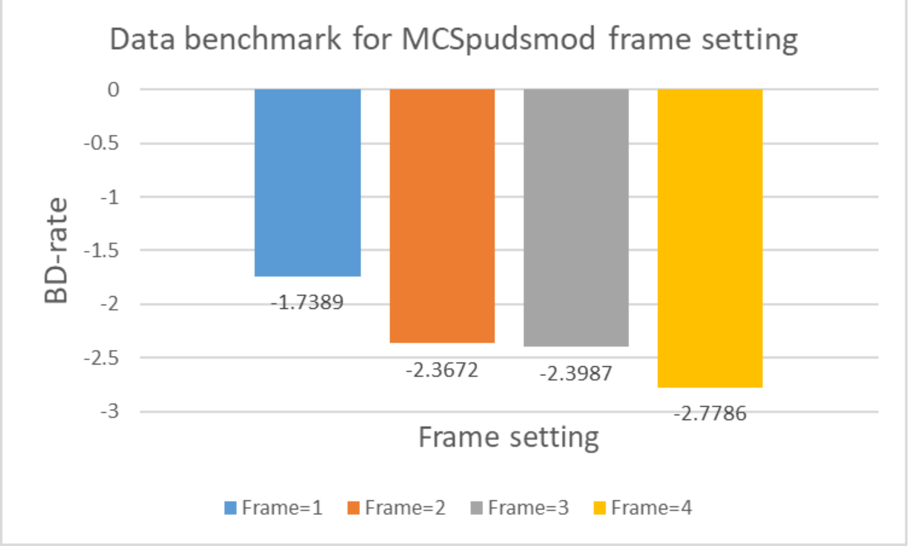

Figure 4. Data benchmark for MCSpudsmod frame setting, strength setting 1 is used.

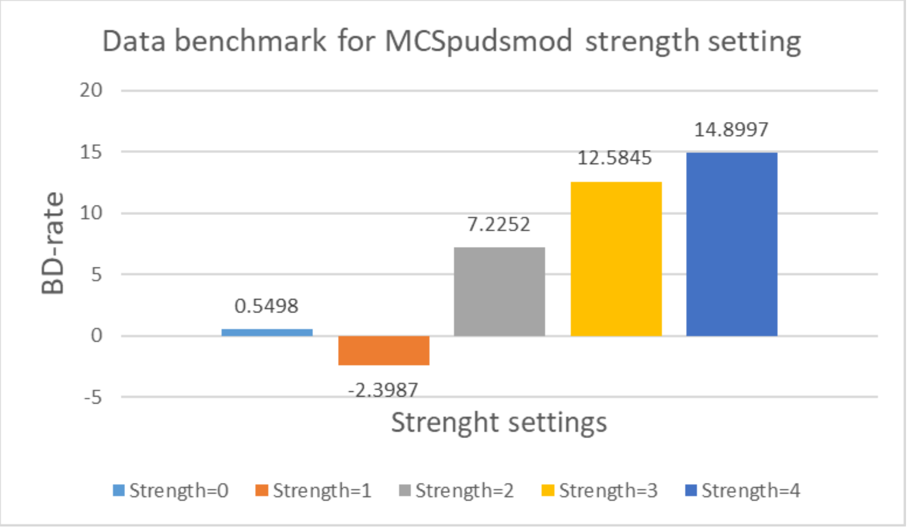

Figure 5. Data benchmark for MCSpudsmod strength setting, frame setting 3 is used.

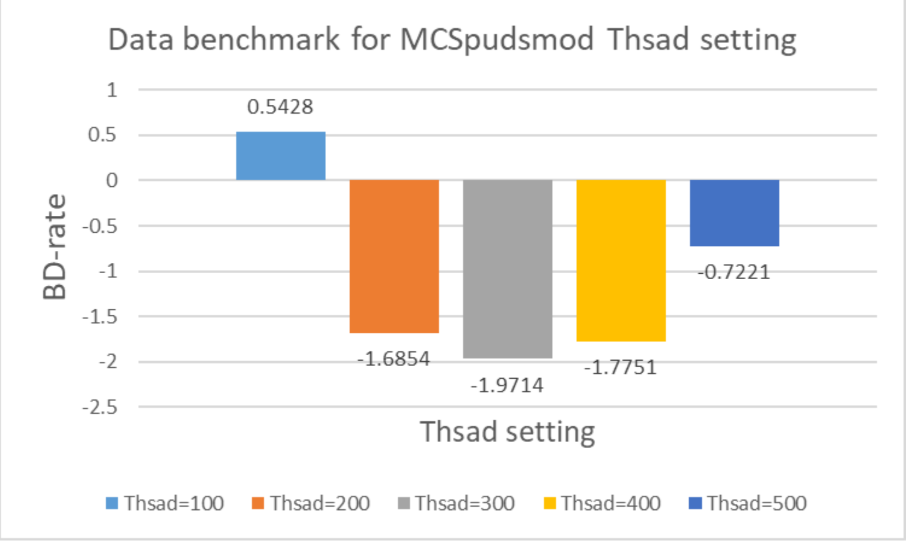

Figure 6. Data benchmark for MCSpudsmod Thsad setting, frame setting 3 and strength 1 is used. A Thsad value of 300 is the default Thsad value for strength setting 1.

Appendix D. Evaluation of NLF identification

Table 11. *Temporal NLF identified on the video ChinaSpeed with synthetic noise added.*

Intensity \ Noise level	1	2	3	4	5	6	7
1	0.9041	1.2871	1.6427	4.423	2.3855	2.8912	3.3037
2	1.6129	2.2941	2.8299	4.0302	3.521	4.0351	4.4271
3	1.7177	2.3474	2.9766	4.3047	4.3065	4.4717	4.868
4	1.3068	1.8974	2.6183	3.9604	4.303	4.5824	5.171
5	1.3902	2.0605	2.706	3.7114	4.0597	4.6234	5.3042
6	1.5866	2.1152	2.7698	3.5906	4.1103	4.7889	5.5718
7	1.8774	2.4243	3.0997	3.7971	4.5005	5.2182	5.9917
8	2.001	2.7321	3.476	3.8971	5.0767	5.7547	6.4216
9	2.0133	2.8226	3.5588	4.0775	5.3082	5.9822	6.6675
10	2.1248	2.9554	3.6368	4.062	5.1471	5.8476	6.4079
11	2.0478	2.8406	3.5146	4.0436	4.7031	5.5131	6.1423
12	1.894	2.5824	3.2261	4.1118	4.3973	5.3298	5.9431
13	1.8826	2.4502	3.161	4.2809	4.3567	5.3264	5.9312
14	1.9666	2.5502	3.2688	4.0893	4.4634	5.3142	6.0593
15	1.907	2.5413	3.1876	4.0498	4.5002	5.4054	6.0604
16	1.9406	2.4467	3.1399	3.9834	4.5221	5.1998	6.0399
17	1.7114	2.2822	2.961	3.8777	4.4375	5.1454	6.0114
18	1.7914	2.394	3.1617	3.4877	4.4873	5.1966	6.0475
19	1.6677	2.3226	3.0473	3.6833	4.6184	5.3899	6.2183
20	1.6843	2.4127	3.1948	4.1158	4.6523	5.4808	6.253
21	1.654	2.4056	3.0901	3.9515	4.5856	5.4751	6.1737
22	1.5704	2.3175	2.9963	3.669	4.3962	5.2034	5.9909
23	1.4285	2.0906	2.8888	3.4901	4.332	5.085	5.7542
24	1.4446	2.2736	2.9875	3.5009	4.3662	5.0788	5.7266
25	1.4536	2.2688	3.0379	3.6056	4.4371	5.1485	5.8541
26	1.4919	2.2393	2.952	3.7221	4.413	5.4121	6.0397
27	1.602	2.3315	3.1117	3.7537	4.5015	5.3529	6.0907
28	1.4534	2.3104	2.9954	3.8792	4.479	5.255	5.9853
29	1.5636	2.2081	2.998	3.5405	4.4518	5.4525	6.075
30	1.5545	2.313	3.0181	3.638	4.4865	5.3254	6.1049
31	1.6339	2.2704	3.0027	3.6865	4.5228	5.1944	6.1598
32	1.5919	2.3845	3.1054	4.0916	4.5318	5.2737	6.0926

Continued on next page

33	1.7042	2.4958	3.2105	3.9398	4.6144	5.483	6.2581
34	1.7188	2.4337	3.1657	3.8161	4.5942	5.6229	6.2939
35	1.5944	2.5625	3.1375	3.8635	4.632	5.6589	6.2692
36	1.8776	2.6159	3.4401	4.4771	4.8952	5.6033	6.5545
37	1.8782	2.4589	3.2702	4.674	5.0059	5.7829	6.6601
38	2.0408	2.7262	3.6231	5.1933	4.7644	5.9655	6.52
39	2.1212	2.6791	3.4593	4.5887	4.8011	5.5973	6.4372
40	1.6635	2.4695	3.2278	4.8942	4.8032	5.7388	6.4282
41	2.4657	3.0526	3.6085	4.4133	5.2899	6.0548	6.8025
42	2.6692	3.2487	3.8885	5.0132	5.3576	6.1919	6.9523
43	2.7603	3.1974	4.2611	4.75	5.5815	6.7221	6.9728
44	1.9728	3.651	4.1458	4.8037	5.1766	6.6027	7.1596
45	2.0876	3.0435	3.8742	4.4438	5.2588	6.2542	7.4234
46	2.8927	3.1552	4.157	4.9051	5.4056	6.9196	7.1643
47	2.9783	3.3993	4.1051	7.4325	5.9355	6.5002	7.6182
48	2.7546	3.9893	4.8421	7.6514	6.459	6.6282	7.83
49	3.2465	3.9333	4.374	7.1119	5.7634	6.8438	7.5343
50	2.5066	3.0282	3.796	5.308	5.7205	6.62	6.8628
51	2.5959	2.949	3.8408	5.1264	5.472	6.429	6.9274
52	1.8645	2.8671	3.9053	4.8584	5.5697	5.9404	7.2764
53	1.5784	2.7052	3.679	3.7635	6.2696	6.2567	7.3033
54	1.6537	2.8139	3.7938	4.3745	6.0529	6.3522	6.7574
55	1.7244	2.7409	3.2776	3.6154	5.0954	5.5182	6.1556
56	1.4202	2.2149	3.0238	3.3576	4.8496	5.5957	6.1387
57	2.4876	3.1858	3.9867	3.5989	5.7802	5.9215	6.7254
58	1.7798	2.2871	3.3556	3.5089	6.6391	6.8443	7.7599
59	2.1607	2.6328	4.2022	3.9986	6.5588	7.3993	9.1534
60	1.6493	4.104	4.3373	5.7678	7.647	7.8567	8.9277
61	1.3664	4.3428	4.136	4.4226	6.6751	8.6228	9.4711
62	1.5187	3.233	4.8902	5.48	6.4505	6.7552	7.171
63	2.0302	2.7022	3.977	8.9326	5.1832	4.6588	5.238
64	0.7736	1.4256	1.8243	6.5343	3.1842	3.0909	3.372

Continued on next page

Table 12. *Spatial NLF identified on the video ChinaSpeed with synthetic noise added.*

Noise level / **Intensity**	1	2	3	4	5	6	7
1	0.5072	0.8856	1.3064	1.7247	2.0601	2.4043	2.7503
2	1.0835	1.7218	2.387	2.9031	3.2576	3.6211	3.9048
3	1.171	1.8115	2.5103	3.2066	3.8575	4.4542	5.1589
4	0.9707	1.6353	2.3248	3.03	3.6941	4.4054	5.0473

Continued on next page

5	1.0296	1.6652	2.344	3.0381	3.7418	4.4337	5.1322
6	1.037	1.706	2.3694	3.0972	3.7834	4.403	5.1811
7	1.0778	1.7203	2.3961	3.0992	3.8111	4.491	5.1768
8	1.0452	1.6578	2.3934	3.1017	3.8093	4.5368	5.2784
9	1.2445	1.846	2.5318	3.3043	3.9579	4.6937	5.5705
10	1.8208	2.2872	2.8299	3.5684	4.1988	4.9305	5.6648
11	1.2407	1.89	2.5934	3.3161	4.0004	4.7657	5.4483
12	0.9224	1.6027	2.3115	3.0803	3.8076	4.4548	5.2846
13	0.9363	1.6204	2.3272	3.0021	3.7423	4.4275	5.153
14	1.1257	1.7261	2.4331	3.1865	3.8629	4.5606	5.1645
15	1.2058	1.7726	2.5113	3.243	3.936	4.5692	5.3558
16	1.2726	1.8071	2.4973	3.2197	3.8845	4.6042	5.3993
17	1.0031	1.6858	2.3449	3.0601	3.7623	4.5147	5.175
18	1.0828	1.7363	2.4198	3.1548	3.8328	4.5059	5.2974
19	1.399	1.9112	2.5369	3.265	4.0146	4.6807	5.4082
20	1.3805	1.912	2.6306	3.2408	3.9465	4.6752	5.3701
21	1.2129	1.806	2.5138	3.2559	3.9332	4.6099	5.3439
22	1.0425	1.7094	2.374	3.1419	3.8613	4.5949	5.3043
23	1.1932	1.79	2.4159	3.1434	3.8695	4.5539	5.2364
24	1.1545	1.8286	2.4756	3.1536	3.8773	4.5505	5.2698
25	1.274	1.8529	2.5501	3.2794	3.9871	4.6699	5.2963
26	0.9642	1.7065	2.5163	3.0954	3.9489	4.74	5.4241
27	1.8003	2.2381	2.8683	3.4726	4.2149	4.882	5.5589
28	1.2726	1.8996	2.5537	3.3131	4.0175	4.7672	5.4598
29	1.7133	2.171	2.7472	3.4633	4.261	4.8079	5.5957
30	1.5062	2.0244	2.6167	3.3739	4.0695	4.7149	5.5307
31	1.8141	2.2514	2.8699	3.6525	4.2232	4.8402	5.632
32	2.4354	2.7812	3.3119	3.8737	4.4475	5.0734	5.6896
33	1.9513	2.3631	2.8871	3.6081	4.3431	4.7898	5.5643
34	1.507	2.0215	2.6937	3.4792	3.9593	4.5773	5.5104
35	1.8176	2.3284	2.8517	3.4574	4.2122	4.808	5.5408
36	2.9114	3.1596	3.579	4.222	4.7849	5.2745	6.1672
37	3.1177	3.4303	3.8547	4.3406	4.9912	5.3238	6.2362
38	2.7229	3.1131	3.5054	3.9746	4.7287	5.3493	5.9592
39	2.9341	3.1705	3.7327	4.1603	4.8136	5.5599	5.8896
40	0.8077	1.582	2.3246	3.1158	3.9071	4.5988	5.5514
41	2.6272	2.9404	3.4173	3.9612	4.6619	5.2823	6.0674
42	3.0508	3.2192	3.6557	4.1558	4.8998	5.4742	5.9618
43	2.2467	2.5591	3.1554	3.7753	4.4422	5.0069	5.8233
44	1.8385	2.2575	2.8277	3.5302	4.2263	5.1234	5.6826
45	1.7848	2.1984	2.9455	3.499	4.1971	4.6579	5.7631
46	1.5083	2.1027	2.7291	3.3571	4.0265	4.5847	5.5616
47	2.1948	2.5403	3.1293	3.751	4.2908	4.8608	5.7911
48	2.3752	2.6307	3.3014	3.7408	4.6134	5.3045	5.9075

Continued on next page

49	2.4238	2.7796	3.2959	3.7556	4.588	5.2618	5.9406
50	2.754	3.0034	3.4474	4.1242	4.5879	5.4019	5.9722
51	2.6476	2.9381	3.3312	3.9729	4.6421	5.1945	5.9593
52	1.9746	2.3352	3.0093	3.5876	4.4595	4.9376	5.8057
53	1.8854	2.3432	3.0675	3.7069	4.2337	4.7037	5.6919
54	1.6996	2.1443	2.7403	3.5858	4.0497	4.815	5.4203
55	1.1927	1.9306	2.4152	3.0733	4.1235	4.712	5.4874
56	1.5837	2.0765	2.6843	3.4067	3.8764	4.6948	5.3219
57	1.5626	2.0551	2.5325	3.2344	4.0943	4.4635	5.3095
58	2.587	2.6087	3.5387	3.9763	4.4743	5.7029	5.5349
59	2.2809	2.4752	3.1654	3.6616	4.4647	5.3271	5.2024
60	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0

Continued on next page

Table 13. *Temporal NLF identified on the video SlideEditing with synthetic noise added.*

Intensity \ Noise level	1	2	3	4	5	6	7
1	0	0	0	13.4375	13.0823	11.9355	13.2908
2	0	0	8.9318	8.0031	9.0899	9.3194	9.4028
3	0	4.7395	4.8346	5.2219	5.8059	6.7536	7.0896
4	1.1517	2.0698	2.8485	3.6119	4.3927	5.1303	5.9945
5	0.8531	1.5598	2.2988	3.1016	3.8952	4.6458	5.481
6	0.8569	1.7889	2.636	3.3394	4.0301	4.6675	5.3558
7	0.8395	1.615	2.4168	3.2171	3.9928	4.7949	5.4524
8	0.8225	1.5739	2.3362	3.0821	3.859	4.6011	5.2581
9	0.8155	1.5832	2.3067	3.0393	3.7262	4.583	5.3098
10	0.848	1.6204	2.4205	3.1367	3.8608	4.5915	5.3209
11	0.8531	1.617	2.4459	3.2378	3.9758	4.6713	5.4437
12	0.8974	1.6755	2.4477	3.2084	4.0329	4.7432	5.5605
13	0.8449	1.6458	2.5122	3.2453	4.0659	4.9056	5.7854
14	0.9128	1.731	2.627	3.3852	4.2921	5.0267	5.8686
15	0.9118	1.7338	2.6458	3.4778	4.2918	5.1248	5.9648
16	0.9572	1.7969	2.8783	3.4945	4.2562	5.0821	5.8892
17	0.9634	1.7884	2.7584	3.4329	4.2585	5.1739	5.9234
18	0.9498	1.8374	2.7496	3.6257	4.3962	5.3215	5.9444
19	0.937	1.8055	2.7682	3.5983	4.4127	5.3015	6.0645
20	0.9354	1.7569	2.6151	3.422	4.2031	5.1243	5.9894

Continued on next page

21	0.9235	1.7399	2.603	3.436	4.2201	5.046	5.9509
22	0.9189	1.7811	2.6912	3.5072	4.3144	5.0759	5.9295
23	0.9506	1.7862	2.6726	3.4948	4.3296	5.1359	5.9444
24	0.9804	1.8385	2.7315	3.5247	4.4034	5.2278	6.0179
25	0.9486	1.8038	2.7297	3.5566	4.4137	5.2873	6.1366
26	0.9424	1.8073	2.7276	3.4975	4.4317	5.24	6.157
27	0.9236	1.7576	2.6877	3.5067	4.3255	5.2338	6.0478
28	0.9019	1.7698	2.6532	3.5244	4.2989	5.1082	5.9382
29	0.9375	1.7922	2.653	3.441	4.1936	5.0499	5.8513
30	0.8991	1.6996	2.5189	3.3546	4.0729	4.9006	5.7902
31	0.9127	1.7318	2.6142	3.3323	4.1514	4.9028	5.7554
32	0.8959	1.6888	2.581	3.2969	4.1892	4.9325	5.7592
33	0.9092	1.7174	2.5832	3.3229	4.2684	5.06	5.9371
34	0.9439	1.7924	2.6706	3.4992	4.2392	5.0919	5.8959
35	0.8847	1.654	2.5231	3.3005	4.1536	4.9993	6.0467
36	0.8921	1.6692	2.5864	3.3316	4.2456	5.1727	6.2157
37	0.9593	1.7865	2.7665	3.4988	4.4416	5.4046	6.2535
38	0.9046	1.7508	2.8353	3.7621	4.6648	5.3914	5.9718
39	0.9776	2.0757	3.1033	3.5244	4.1075	4.6537	5.2306
40	0.7423	1.432	2.1651	2.6653	3.3352	4.0678	4.7142
41	0.9068	1.5904	2.2419	2.8019	3.4576	4.1078	4.8354
42	0.9689	2.01	3.168	3.8151	4.3982	4.9723	5.6156
43	0.9701	1.8071	3.0186	3.9441	5.1381	5.92	6.6626
44	0.9627	1.8188	2.8507	3.6893	4.79	5.911	6.9165
45	0.9957	1.953	2.9108	3.6793	4.547	5.4733	6.5715
46	0.9524	1.7364	2.6451	3.4191	4.3118	5.2715	6.3108
47	1.0265	1.9351	2.8485	3.5448	4.515	5.5264	6.6209
48	0.9353	1.7845	2.8686	3.9118	5.0761	6.0346	6.7738
49	0.9136	1.8179	3.1877	4.1418	4.9412	5.6146	6.2446
50	1.2017	2.0496	2.7402	3.3033	3.9592	4.7152	5.4875
51	0.8086	1.4512	2.1406	2.8133	3.5595	4.322	5.2084
52	1.0308	2.072	2.8802	3.462	4.1259	4.8954	5.6511
53	0.9289	1.7783	2.8596	3.9464	4.9639	5.6747	6.3919
54	0.8949	1.7721	2.9041	3.856	4.7695	5.6089	6.4831
55	1.0559	1.9064	2.5978	3.2255	4.0814	5.1645	6.2745
56	0.8013	1.463	2.1846	3.09	4.1929	5.3097	6.2322
57	1.0169	2.1904	3.3222	4.1914	4.8877	5.417	5.9253
58	1.1451	2.2947	3.0274	3.5531	4.106	4.6412	5.1868
59	0.7888	1.3832	1.985	2.6167	3.3007	4.0172	4.7137
60	1.6349	2.166	2.7239	3.2529	3.8206	4.4242	5.1052
61	0	5.1316	5.2754	5.4634	5.6867	6.0265	6.5597
62	0	0	8.6231	8.8053	8.7564	8.7432	8.8562
63	0	0	12	12.5634	12.4462	12.2115	11.915
64	0	0	0	0	16.3875	16.0965	15.7856

Continued on next page

Table 14. *Spatial NLF identified on the video SlideEditing with synthetic noise added.*

Noise level / Intensity	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	10.272	0	20.302	0
4	0.7183	2.2867	5.0777	7.8578	12.2976	18.9857	23.9607
5	0.7696	2.2052	4.8048	8.0065	13.019	19.0997	24.4141
6	0.6822	2.29	4.8658	8.1441	12.3917	18.513	25.1809
7	0.6988	2.3103	4.6552	7.9856	12.5661	18.3478	23.8088
8	0.8037	2.4304	5.001	8.4753	12.7897	18.3609	24.7921
9	0.8157	2.3784	4.7743	8.5891	13.0991	17.9108	26.0727
10	0.9658	2.5921	4.8897	8.9172	13.3614	19.6237	26.0171
11	0.8782	2.6839	5.1023	8.9089	12.7519	19.7041	26.6612
12	0.8622	2.5261	5.1315	8.6094	13.5318	18.8181	25.8732
13	1.1846	2.8301	5.3567	9.021	13.782	19.2793	25.5001
14	1.0888	2.6329	5.5591	9.341	13.8784	20.3418	26.8147
15	1.1293	2.8919	5.5819	9.335	13.9015	19.7002	26.6274
16	1.1461	2.8971	5.3681	9.2067	13.4835	20.1827	26.5609
17	1.2622	3.0064	5.826	9.4723	13.6087	20.7587	27.3159
18	1.2996	2.9992	5.4732	9.4698	14.2396	21.0235	27.387
19	1.0535	2.7118	5.4567	8.9394	14.1028	20.4881	26.8518
20	0.9827	2.6424	5.4969	9.3813	14.165	20.3255	26.061
21	1.4092	3.2605	6.4283	9.6391	15.2834	21.5433	27.7717
22	2.0951	3.9932	6.9102	10.9487	15.991	22.5229	29.3564
23	1.9349	3.4273	6.2784	10.8134	15.0938	22.0278	28.7632
24	1.9022	3.6328	6.6077	10.6613	15.1356	22.3252	29.2996
25	1.7933	3.4531	6.1701	10.684	15.828	21.7443	28.7924
26	1.825	3.4915	6.3552	10.2863	15.382	21.5951	27.9472
27	1.9299	3.7629	6.5508	10.4936	15.9176	21.4482	29.5091
28	1.7113	3.6832	6.6582	10.2777	15.6594	22.3843	30.2109
29	2.0592	3.6877	6.6944	10.9008	15.6312	21.6052	29.9853
30	1.8675	3.5954	6.4748	10.592	15.2564	21.3899	28.5352
31	1.9299	3.5161	6.5081	10.6753	15.1514	21.8829	29.5626
32	2.3227	3.8931	6.8351	10.7396	15.7294	22.7313	29.0508
33	2.8274	4.5054	7.2581	11.2128	16.4011	23.2048	29.8006
34	1.9347	3.6927	6.7003	10.4553	16.5034	22.2573	29.3789
35	2.0795	3.669	6.7555	10.3869	15.6695	21.4104	28.9353
36	3.224	4.8909	7.8641	11.9771	16.2962	23.806	29.4991
37	3.721	5.4247	8.0657	12.7645	17.8152	24.478	30.9268
38	2.3494	4.2148	7.3056	11.2327	16.3979	22.5682	29.5174
39	0.5182	2.0633	4.6657	8.0147	12.5081	18.4931	24.6246

Continued on next page

40	2.8244	3.8347	5.4604	8.9106	13.4987	19.1569	25.2218
41	2.9529	4.8193	7.4851	12.2712	17.8531	24.0189	31.8458
42	3.5234	5.2173	8.2356	12.4478	18.3612	24.4743	32.5748
43	3.4743	5.1331	8.203	12.3932	18.4085	24.9647	33.6218
44	3.1928	4.7845	8.308	12.1453	17.921	26.129	33.2781
45	3.6292	5.4594	8.8944	13.084	19.278	25.2454	34.061
46	9.0504	10.8474	12.3983	19.1098	22.2475	28.5018	41.882
47	2.6162	4.4571	7.5268	12.0348	17.7126	23.6851	33.1014
48	17.861	21.8141	21.8031	26.5513	33.6012	39.0948	44.286
49	6.7152	7.9214	11.7012	15.9065	22.4824	29.3165	38.6079
50	0.6781	2.5958	5.6952	9.7187	14.9512	21.9286	29.5689
51	3.6963	5.3773	8.3721	12.8159	18.6987	24.7581	33.1613
52	4.4233	6.1199	9.2617	13.8122	19.2832	25.8101	34.6264
53	18.6781	21.5668	24.5905	29.2787	36.2821	42.1401	51.7306
54	2.3816	3.9613	7.3888	11.6724	17.244	23.8034	32.0864
55	0.542	2.1737	4.798	8.6428	13.0719	19.0991	25.2586
56	24.0878	25.286	28.0816	30.6016	35.8698	41.4623	48.7722
57	19.4628	19.7762	22.2346	26.7905	33.5569	38.2959	49.6639
58	0.5128	2.0086	4.5412	7.9786	12.4318	17.7307	23.9443
59	0	0	4.5844	8.24	11.7946	16.8146	24.8624
60	0	0	0	0	0	0	0
61	0	0	0	0	0	0	0
62	0	0	0	0	0	0	0
63	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0

Continued on next page

Appendix E.
PSNR vs noise level

```
procedure
    noise_level ← 3
    h ← 1000
    w ← 1000
    size ← h ∗ w
    peak_value ← 255
    MSE ← 0
    for w = 1 : w do
        for h = 1 : h do
            MSE ← MSE + normal_random_value(−1, 1) ∗ noise_level
    MSE ← MSE / size
    PSNR ← 10 ∗ log10(peak_value^2 / MSE^2)
```

Algorithm 2: Algorithm to measure the PSNR for a fixed noise level (noise level 3) on a image with bit death 8.