Problem Description

JPEG2000 is the dominant coding system for transporting video over IP networks due to its high quality compression capabilities and advanced functionalities. JPEG2000 inherent scalability [1] functionalities may be exploited to adapt the video bitrate to the conditions in the network. The rate control in such a system adjusts the bitrate by extracting a certain number of quality layers from the video stream. In "Adaptive Rate Control of Motion-JPEG2000 Video over IP Networks" by Rong-Yu Qiao and Michael H. Lee, a system using JPEG 2000 quality scalability is used for such a purpose. This thesis should investigate further into the use of JPEG 2000 quality scalability for a proposed scenario using IP networks. Questions to be answered are the number of quality layers to be embedded in the video stream. Is there a preferred number? It should also look into the subjective perception in such a system by performing subjective test and suggest future design of a smart rate control.

[1] "ITU-T T.800 JPEG 2000 Part 1: Core Coding System" will be the main source of information on JPEG 2000 quality scalability.

Abstract

In this thesis, the JPEG 2000 quality scalability feature was investigated in the context of transporting video over IP networks. The goals of the investigation was two-fold. First, it was desired to find a way of choosing the number of quality layers to embed in a JPEG 2000 codestream. In previous work, this choice has been more or less arbitrary. Second, it was desired to find how low the video bitrate could be dropped before it became perceptible to a viewer. This information can be used in an IP networking scenario to e.g. adapt the video bitrate blindly according to the measured channel capacity as long as the drop in bitrate is expected to be imperceptible. When the drop in bitrate is expected to be perceptible, a switch could be made to a smoother bitrate adaptation.

A way of choosing the total number of quality layers to embed in a codestream was found by minimizing the difference in predicted quality between direct and scaled compression. Scaled compression is the compression which is achieved by extracting quality layers. The minimization procedure was bound by the speed of the encoder, as it takes longer for an encoder to embed more quality layers. It was found that the procedure was highly dependent on the desired bitrate range.

A subjective test was run in order to measure how large a drop in video bitrate had to be for it to become perceptible. A newly developed JPEG 2000 quality layer scaler was used to produce the different bitrates in the test. The number of quality layers to embed in codestream was found by using the minimization procedure mentioned above. It was found that, for the bitrate range used in the test, 2 - 30 Mbits/s for a resolution of 1280x720 at 25 frames per second, the magnitude of the drop in bitrate had to be at least 10 Mbits/s before the participants in the test noticed it. A comparison with objective quality metrics, SSIM and PSNR, revealed that it was very difficult to predict the visibility of the drops in bitrate by using these metrics. Designing the type of rate control mentioned in the first paragraph will therefore have to wait until a parameter with good predictive properties can be found.

Sammendrag

I dette prosjektet har kvalitetsskalerbarhet i JPEG 2000 blitt undersøkt innenfor rammene av IP nettverk. Målet med prosjektet har vært todelt. Først var det ønsket å finne en fremgangsmåte for å velge hvor mange kvalitetslag som bør bygges inn i en JPEG 2000 kodestrøm. I tidligere arbeid synes dette valget å være vilkårlig. I tillegg til dette var det ønsket å undersøke hvor lavt en videobitrate kunne falle før det ble observerbart. Denne informasjonen var tiltenkt bruk i et IP nettverks-scenario, hvor man kan tilpasse videobitraten blindt etter kanalens kapasitet så lenge fallet i bitrate forventes å være umerkbart. Når fallet i bitrate forventes å være merkbart kan man bytte til en jevnere form for tilpasning av bitraten.

En fremgangsmåte for å velge antall kvalitetslag som skulle bygges inn i kodestrømmen ble funnet ved å minimere forskjellen mellom forventet kvalitet i direkte og skalert kompresjon. Skalert kompresjon er herved navnet på kompresjonen som oppnås ved å ekstrahere kvalitetslag. Minimeringsprosedyren var begrenset av hastigheten til enkoderen. Dette var fordi det tar lengre tid å enkode en kodestrøm med mange kvalitetslag. Det ble også funnet at prosedyren var høyst avhengig av den ønskede størrelsen på bitrateområdet.

For å måle hvor lavt et fall i bitrate kunne være før det ble merkbart ble det kjørt en subjektiv test. En JPEG 2000 kvalitetsskalerer utviklet under prosjektet ble brukt til å generere de forskjellige bitratene som ble brukt i testen. Antall kvalitetslag som skulle bygges inn i kodestrømmen ble funnet ved hjelp av minimeringsprosedyren beskrevet ovenfor. Fra testen ble det funnet at, for bitrateområdet som ble brukt i testen, 2-30 Mbits/s ved en oppløsning på 1280x720 ved 25 bilder i sekundet, krevdes det et fall på minst 10 Mbits/s før det ble merkbart. En sammenligning med objektive kvalitetsmål, SSIM og PSNR, viste at det var vanskelig å forutse synligheten til fallene i bitrate ved bruk av disse målene. Å utvikle en type ratekontroll som beskrevet i første avsnitt blir dermed satt på vent inntil en bedre måleparameter finnes.

Contents

Chapter 1

Introduction

In broadcasting, efficient methods for transporting video are of utmost importance. A typical scenario might be the transportation of video from a sports event to a studio. A relatively new way of accomplishing this is via IP networks, which is a naturally flexible solution. Such a network can be dedicated to the task of transporting video, or it can be a more general purpose network such as the Internet. The latter has not been extensively used, mainly because it is not as reliable as a dedicated network. It is, however, a much cheaper solution than having to set up a dedicated network. If the Internet proved a reliable medium for video transport, it would result in enormous savings in both time and money for the broadcaster. In the master thesis "Reliable broadcast contribution over the public internet" [2] currently being written by M. Markman and S. Tokheim, the viability of video transportation over the Internet is investigated. The evidence suggests that the Internet may provide a reliable medium for video transportation.

Should one decide to use the internet for video transportation, there are some practical questions that need to be answered. For example, when the network gets congested, how should it be handled? This is one of the questions that motivates this thesis. One solution is to use the JPEG 2000 [1] quality scalability feature to adapt the video bitrate to the conditions on the channel. Scalability avoids transcoding the video, and is much less complicated to implement than a full encoder and decoder. This makes the adaptation process cheap both in terms of implementation and in terms of processing. To use scalability, the video must already be encoded, and later, decoded. Solutions in this regard already exist. An example is the TVG450 produced by T-VIPS, which handles encoding and decoding of JPEG 2000 images, and is designed for use in IP networks. JPEG 2000 is a good solution in a broadcasting scenario because it only uses intra-frames (JPEG 2000 is just an image codec [1]. When it is spoken of as a video codec, it just means a concatenation of still images). By using only I-frames, it is ensured that errors won't propagate to future frames. In addition to this, the delay produced by an intra-frame codec is only the processing time required to process the frame itself. In this project, JPEG 2000 is further suitable because of the inherent scalability properties of the codec.

In this thesis, the quality scalability feature of JPEG 2000 is investigated. Quality scalability is a method for embedding several different video qualities (or bitrates) in the same image bitstream. Ultimately, the goal is to use this feature in video transport to adapt to the current conditions on the transportation channel. By doing this, one can adapt the video bitrate, without transcoding, to situations where the channel is partly blocked (e.g. packet drops in the internet). Similarly, the video bitrate could be scaled down at transmission path nodes connecting to other transmission paths of unequal expected capacities. Some research on this subject has already been done. In a paper by R. Qiao and M. Lee [3], the JPEG 2000 quality scalability feature was used to adapt to a network bitrate, by measuring the TCP friendly rate, and by using the video transportation protocol (VTP) [4]. However, some questions remain unanswered. The first question is "How many quality layers should be embedded in the image bitstream?". It seems that, without loss, one would like an infinite number of quality layers, thereby achieving infinite granularity. A second question is "Should the bitrate be directly adapted to the conditions on the channel, or is it a better idea to have a smoother variation in bitrate?" One solution might be to switch between these two ideas based on how they affect the perceived quality.

In order to answer the questions mentioned, a JPEG 2000 quality layer scaler will be implemented. This is a software program which takes a JPEG 2000 codestream, and extracts a number of quality layers in order to achieve a certain image bitrate. This tool will be used in the development of a procedure for choosing the total number of quality layers to embed in a JPEG 2000 codestream. Using the results from this procedure, and the JPEG 2000 scaler, a subjective test will be carried out in order to investigate how low the video bitrate can be dropped before it becomes perceptible. These subjective evaluations will be compared to objective quality metrics. It is desired that the knowledge gained from these experiments will be useful on the way to creating a fully functional channel-adaptive video scaler, using JPEG 2000.

Organization of the thesis

The thesis will first cover some theory needed to understand the rest of the report. First it will explain the use of objective quality metrics. Second, it will provide a quick overview of the JPEG 2000 image coding system, and a more thorough look into the scalability feature. Following the theory section, an overview of previous work will be provided. This chapter will look into what has been done in the field before, and more importantly, what has not been done. This will motivate the next chapter, the problem statement. Here the question that the thesis aims to answer will be defined. The methodology chapter will explain how the JPEG 2000 scaler was made, how it was used to find the preferred number of quality layers to embedd in the image bitstream, and lastly, how a subjective test was run in order to investigate the effect of drops in bitrate. The results and discussion will present the results, and compare them to objective quality metrics. In the conclusion, the discussion is summarized, and the problem statement is answered. Future work is suggested.

The report will also contain a list of references, and an appendix. The appendix will contain images, figures, tables, and other details that was left out of the rest of the report. The source code used throughout the project will be provided in a zip file. In this zip file, readme.txt files are provided in the different folders for explanation of the content.

Chapter 2

Theory

2.1 Objective image quality

Ultimately, subjective image quality is the only thing that matters. However, measuring subjective quality is time consuming and difficult, therefore objective measures are often necessary. For example, whenever an image or video codec performs rate-distortion optimization, it has to calculate the image distortion using an objective metric. The optimality of such a procedure is therefore only optimal in the context of the metric it uses. Here, two different objective metrics will be discussed: the PSNR and the SSIM index [5]. Both of these are so-called full reference metrics, as they work by comparing one image against another. In contrast, no reference metrics tries to measure the image quality by only looking at the one image. Full reference metrics are much more tried and tested, and are often the natural choice.

2.1.1 PSNR

PSNR (Peak signal-to-noise ratio) can be used as a measure of the difference between images. It is defined by equation 2.1.

$$PSNR = 10 \log_{10} \frac{I_{max}^2}{MSE} \tag{2.1}$$

Where I_{max} is the dynamic range of the image (for an 8-bit image, the dynamic range is 255). Relating it to the dynamic range means that the metric can be compared when used on images of different dynamic ranges. Taking the logarithm compresses the range of values down to a more readable range. However, the information contained in the PSNR is actually no more than that of the MSE (mean squared error).

The problem with PSNR is that MSE is not a good measure of how much distortion there is in an image when passed through the human visual system (HVS) [6]. The MSE does not account for the fact that the human brain is very good at extracting the useful information in an image, and filling in the blanks. For example, the HVS detects noise more easily in smooth regions than regions with a lot of textures (frequency masking). The HVS is not very sensitive to small contrast changes, relative luminance, spatial shifting or rotation. The MSE is thus a pure mathematical measure of how similar two signals are, and it is used primarily because it is easy to calculate and implement.

Of course, there exist methods for improving the PSNR. For example by weighting the MSE according to how sensitive the HVS is to errors in different spatial frequencies [6]. This is described by the contrast sensitivity function. Another common improvement is to only evaluate the PSNR on the luminance component of an image, as the HVS is much more sensitive to errors here than in the color components.

2.1.2 SSIM

The SSIM (Structural SIMilarity) index uses a different approach from the PSNR and its siblings [5]. The SSIM index tries to mimic the HVS, in that it tries to compare the difference in structural information between two images. In contrast, the PSNR-type methods measure the accumulated errors between the two images. SSIM is therefore not as sensitive to things like contrast and relative luminance, as long as the structural information is still present. SSIM is still sensitive when it comes to spatial shifting and rotations, but in the context of image and video coding, this is rarely a problem. Even so, methods have been developed that accounts for this as well [6]. An SSIM index value always lie between 0 and 1, where 1 only happens when the images are equal. The SSIM index is also symmetric, so that it doesn't matter which one is the reference image and which one is the degraded one. If the SSIM index has been calculated on two images, and one wishes to compare the two values, it is tempting to assume that the values are linear, e.g. such that a value of 0.99 is 1% better than 0.98. This is not necessarily the case, as can be verified by experimentation. It is difficult to find information about this in the literature, but by experimentation it seems that the DSSIM (structural dissimilarity) makes for a better comparison (see equation 2.2). In this equation, the SSIM values are read as "how far is this value from the reference".

For example, the value 0.98 is twice as good as 0.96, since it is twice as close to 1. For comparisons, equation 2.3 may therefore be used.

$$DSSIM = \frac{1}{1 - SSIM} \tag{2.2}$$

$$compare12 = \frac{1 - SSIM(reference_2, image_2)}{1 - SSIM(reference_1, image_1)} \tag{2.3}$$

Equation 2.3 gives the ratio of the distance between the reference and the impaired version of image 1 compared to image 2. Regardless of the actual values produced by equation 2.3, the fact that one SSIM value is higher or lower than the other is preserved through the equation.

2.2 About bitrates

By bitrate, it is meant how many bits that are handled per second: $\frac{bits}{second}$. In video, the bitrates are often in the order of millions, so the short form Mbits/s is used. This simply means $\frac{1000000bits}{second}$. It is easy to forget that a bitrate is not a compression factor. As an example, a video clip at 1920x1080@25fps, with three color components (e.g. RGB), and 8 bits precision, requires a bitrate of $\frac{1920*1080*3*8*25bits=1244Mbits}{second}$. Often, the bitrate available is much lower, so the video has to be compressed. For 30 Mbits/s, for example, the video has to be compressed with a factor of 30 / 1244 = 0.024. If the video to be compressed had a lower resolution, for example, 1280x720, the compression factor would be $\frac{1920*1080}{1280*720} = 2.25$ times smaller than this. This is important to keep in mind later in the report, as different resolutions are used. The bitrates discussed are always related to the resolution and number of frames per second (fps).

2.3 A summary of the JPEG 2000 coding system

A JPEG 2000 [1] image is coded in several steps. First the image is preprocessed. Here the image is converted to YC_bC_r or YUV color space. This can be done either reversibly or non-reversibly. The image may also be divided into tiles (a spatial subdivision of the image), which can be handy if memory is limited. After preprocessing, the different color components are transformed by a wavelet transform, dividing the image into spatial frequency subbands. This process is comparable to filtering the image with a filterbank. The transform may be reversible or non-reversible. The wavelet coefficients resulting from the transformation may be quantized, which is a non-reversible operation. The final coding step is EBCOT (embedded block coding with optimal truncation). It is here that compression is achieved. This is done by using an arithmetic entropy coder. EBCOT is also responsible for generating the highly flexible and scalable JPEG 2000 bitstream. In such a bitstream, several versions of the same image may be embedded. The next chapter discusses this feature in detail.

In order to convey information about the coding choices, a JPEG 2000 image is wrapped in what is called a codestream [1]. The codestream consists of a main header, and tile headers preceding the bitstream contained in each tile. Information in a tile header always overrides information in the main header. Information in the headers are contained in marker segments, which are signalled by different types of markers. Some markers also exist inside the bitstream, such as the start-of-packet marker.

The JPEG 2000 codestream syntax can be quite complicated, and the reader is referred to [1] for more information.

2.4 Scalability in JPEG 2000

In the context of image coding, scalability is a way of packaging several versions of an image within the same codestream. In JPEG 2000, a single codestream can contain different numbers of resolutions, qualities, components, and positions, which can all be extracted without decoding the image. This chapter contains a detailed description of the scalability feature in JPEG 2000. More information can be found in [1].

2.4.1 Different types of scalability in JPEG 2000

There are four types of scalability in JPEG 2000; quality, resolution, component, and spatial scalability. Quality scalability is achieved by encoding the image data as a bitstream consisting of several layers. By extracting such layers from the bitstream, the quality of the decoded image is enhanced progressively. The other types of scalability works in a similar way. For instance, the resolution of an image may be increased progressively by extracting more resolution levels. Component scalability usually refers to the ability to progressively add color to an image. In the YUV color space, for instance, the luminance component may be extracted in order to obtain a black and white image. The chroma components may then be progressively added in order to put color in the image. In spatial scalability, the different physical parts of an image is obtained progressively. For instance, it is possible to extract the top half of an image only, and obtain the lower half by extracting the relevant part of the bitstream.

2.4.2 Progression orders

Scalability can be achieved by means of five different progression orders:

- LRCP (layer - resolution level, component, position)
- RLCP (resolution level, layer, component, position)
- RPCL (resolution level, position, component, layer)
- PCRL (position, component, resolution level, layer)
- CPRL (component, position, resolution level, layer)

A progression order describes how the compressed image data is ordered in the bitstream. As an example, in order to read a bitstream with the LRCP progression order, one could use the pseudo code shown below. See also figure 2.1 for further illustration.

Algorithm 1 Pseudocode for reading packets in bitstream

```
for all layers do
    for all resolution levels do
        for all components do
            for all positions do
                read packet containing information about the current layer, resolution level, component, and position
            end for
        end for
    end for
end for
```

These for-loops can be inter-changed to match the different progression orders. In principle though, any type of scalability can be achieved regardless of the progression order, but it is easier to implement e.g. quality scalability, with the LRCP progression order, since the packets in the bitstream contain all the lower layers first. A closer look at the bitstream packets is given in section 2.4.3.

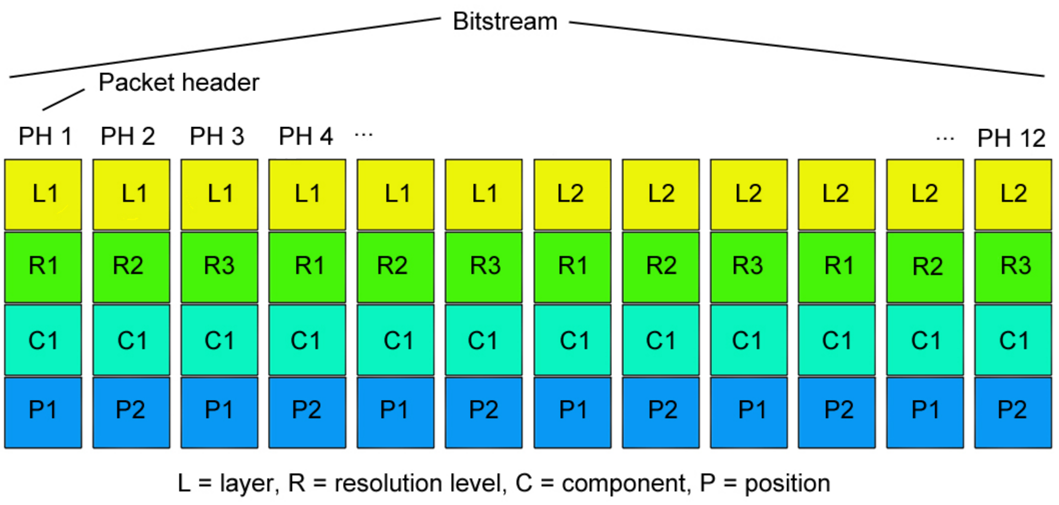


Figure 2.1: Illustration of a JPEG 2000 bitstream using the LRCP progression order. Each vertical group of blocks illustrates a packet. In this example, there are two layers, 3 resolution levels, 1 component, and 2 positions. To extract the first layer, one need only extract the first 6 packets.

2.4.3 How scalability is implemented in JPEG 2000

In this section, some details of the implementation of the scalability features of JPEG 2000 will be provided.

EBCOT

JPEG 2000 uses the EBCOT (embedded block coding with optimal truncation) scheme in order to perform entropy coding and organize the bitstream. This technique is applied after the wavelet subband decomposition and quantization, and works by dividing the subbands into equally sized codeblocks, which are coded independently. Each codeblock is coded by a bitplane encoder, coding the most significant bits first. The entropy coding is done by an MQ arithmetic encoder.

To perform rate allocation, the encoder decides how much each codeblock should contribute to a subband according to how much distortion it would remove. This is done until the specified total bitrate is achieved. Put differently, the encoder is deciding on ”truncation points”. When this is done optimally in a mean-square-error sense, it is said that ”optimal truncation” is reached. If more than one layer is used, the encoder has to decide on optimal truncation points for each of the layers.

After the optimal truncation points has been found, the encoder has information on how much each codeblock should contribute to the total bitstream, along with how each codeblock relates to the different combinations of progression types. The encoder can then assemble packets in the bitstream by accepting the most contributing codeblocks first, and by grouping them according to which combination of layer, resolution level, component, and position they belong.

Implementation of the progression types

How is it that layers can yield higher and higher quality when more of them are included, and how are they related to the codeblocks generated by the EBCOT scheme? Below is a more detailed description of the different progression types.

- **Quality layers**

  For a given tile, a layer contains a certain number of consecutive bitplane coding passes from each codeblock within that tile. The bitplane coding is what makes layers represent different qualities, as each layer receives a varying degree of pixel accuracy (i.e. significant bits).

  The exact number of coding passes included for a given codeblock in the layer depends on the rate allocation method, but in general, the information contained in a layer is rate-distortion optimal in a PSNR sense. This means that extracting a whole number of layers from a codestream yields an image with the best possible PSNR for the resulting image size (not counting meta informational overhead, such as headers).

- **Resolution levels**

  A resolution level consists of a number of subbands generated by the wavelet transform. The lowest resolution level consists of the LL band, while the higher levels adds the contribution from the HH, LH, and HL bands as well. The LL band represents the version of the image which is low pass filtered in both the horizontal and vertical direction, while HH represents the high pass version. The LH band is first low pass filtered in the horizontal direction, and then high pass filtered in the vertical direction. The HL band is the opposite. These subbands are split into codeblocks, creating the link between codeblocks and resolution levels.

- **Components**

Each color component contains its own set of resolution levels. This means that, since a codeblock is a subsection of a subband (and therefore a resolution level), the codeblock also has information specific to the given component.

- **Positions**

  A position is directly translatable to a precinct. A precinct is a spatial subsection of a resolution level. This means that a precinct is actually a collection of codeblocks within a resolution level.

Bitstream packets

The codeblocks are arranged in the codestream according to the given progression order. This is done by grouping the compressed image data into packets. Each packet contains codeblocks which contain compressed image data for the current layer, resolution level, component, and position. See figure 2.1 for an illustration of the packets.

Since it is possible for a packet to contain the same codeblocks as a previous packet, a packet header specifies whether a codeblock is included in this particular packet or not. In addition, whenever a packet includes no codeblocks at all, this is signalled by a single bit in the packet header [1]. However, it is important to note that packets are always byte-aligned, so the minimum size of a packet is always one byte. For more information about packet headers, the reader is referred to [1]. In addition to the packet header, it is possible to specify the usage of a start-of-packet (SOP) marker. A SOP marker is placed just before each new packet in the bitstream. It was originally intended to be used for error correction, but it can also be used to bypass the decoding of packet headers entirely. Each SOP marker requires 6 bytes of space [1], so a certain cost is unavoidable.

Extracting a layer

With a codestream as described, it is possible to extract different amounts of quality layers, resolutions, components, and positions, without decoding. For example, extracting a layer from a codestream amounts to reading the main and tile-part headers for information, and, for the bitstream in each tile-part, discarding all the packets belonging to a higher layer than what is specified. The main and tile-part headers are then updated to reflect the changes in the bitstream. In principal, this is a simple task, but is complicated by other features which purpose is to obtain codestream flexibility. An example of a complicating feature is the ability to change the progression order in a JPEG 2000 codestream.

2.5 Differences between direct and scaled compression

In this section the goal is to examine some of the differences between direct and scaled compression. Scaled compression is the compression achieved by extracting a number of quality layers from a codestream, while direct compression is regular compression without the use of scalability.

When comparing direct compression with scaled compression there are essentially two parameters affecting the result: the bitrate range, and the number of quality layers used. In creating a quality-scalable JPEG 2000 codestream, it is necessary to specify both these parameters. In practice, this is done by specifying a number of rates which the bitstreams will include as layers. For example, specifying the rates 0.2 and 0.3 will yield two layers, where the first layer contains the image compressed with a compression factor of 20%, and the second layer contains enough information to get a compression factor of 30% when combined with the first layer. This process adds some overhead, since more packets, and therefore packet headers (and possibly start-of-packet markers), must be produced. (see chapter 2.4.3 for a description of JPEG 2000 packets).

This brings up a number of limitations. First of all, one cannot extract a quality layer with a lower bitrate than what was specified. The same goes for higher bitrates. Second, since it is only possible to extract an integer number of quality layers from the bitstream, it is therefore not possible to extract a rate which lies in between two layers. Third, there is a limitation on the number of layers one can use, since more layers means more overhead and longer encoding times.

Chapter 3

Previous Work

In this chapter, the aim is to give a quick overview over what has been done with JPEG 2000 quality scalability in video transport applications.

Rong-Yu Qiao and Michael H. Lee have contributed several papers in the area [3][7]. In "Adaptive Rate Control of Motion-JPEG2000 Video over IP Networks" [3] they use Motion JPEG 2000 to transport video in an IP network. A self-made JPEG 2000 stream scaler [7] is used to adapt the video bitrate to the current conditions on the channel, i.e. quality layers are extracted from the Motion JPEG 2000 codestream such that the bitrate is lower than the capacity of the channel. A maximum of 8 different bitrates can be extracted with their stream scaler [7]. The current capacity of the transport channel is estimated by the TRCP (TCP-Friendly Rate Control) and VTP (Video Transport Protocol). The conclusion of the study is that by adapting the video bitrate with the stream scaler, they obtain perceptually good results, and the ability to gracefully degrade the video.

There are a number of issues with the study. First of all, the number of layers used, eight, seems rather arbitrary. In principle, one would want as many quality layers as possible in order to have finer granularity. Second, although they report perceptually good results, they have only measured the PSNR of the video. As discussed in the theory section of this report, PSNR is not necessarily a good measure of subjective quality.

Lastly, the use of the TRCP protocol is based solely on measuring the IP network conditions, and although that might give a good indication of the situation on the channel, it says nothing about the subjective quality of the video. The VTP protocol [4] is similar in that it is calculated only based on the conditions on the channel. VTP is designed to work with wireless networks, and tries to adapt more smoothly than TRCP. This is possible because it discriminates between errors in transmission and errors because of congestion (TRCP treats everything as congestion).

However, neither VTP, nor TRCP, uses information about how visible their bitrate adaption is to the viewer of the video.

Chapter 4

Problem Statement

In this thesis, the aim is to investigate some practical aspects of the JPEG 2000 scalability feature in video over IP applications. This is interesting mainly because the literature in the area is scarce, and not very in-depth. After all, JPEG 2000 is mainly an image codec, not a video codec. As described in the "Previous Work" chapter, some studies have been done using JPEG 2000 scalability in video transport over IP, but several questions remain unanswered, and subjective results are hard to come by. This thesis will try to provide answers that are missing from previous work.

To put the thesis into context, a scenario is used where the goal is to transport video over IP networks, with a JPEG 2000 scaler sitting at a node in the network, adapting the video bitrate to the conditions on the channel(s) following the node. In figure 4.1, such a scenario is illustrated. Here, the video is encoded once, sent over a network with high capacity, such as a dedicated network, and then adapted to three other smaller networks according to their measured capacities. Since the JPEG 2000 scalability feature is responsible for the bitrate adaption, transcoding is not necessary. This can be a huge time saver, and less costly to implement, since a JPEG 2000 scaler is much less complex than a full encoder and decoder.

The thesis will seek to answer two questions that is of interest in the scenario described above.

1. How should the number of quality layers in a scalable JPEG 2000 codestream be chosen? This is an interesting question because, if there were no penalties, one would like to have as many quality layers as possible in order to get a fine granularity.

2. How much can the video bitrate drop before it becomes perceptible? If a metric

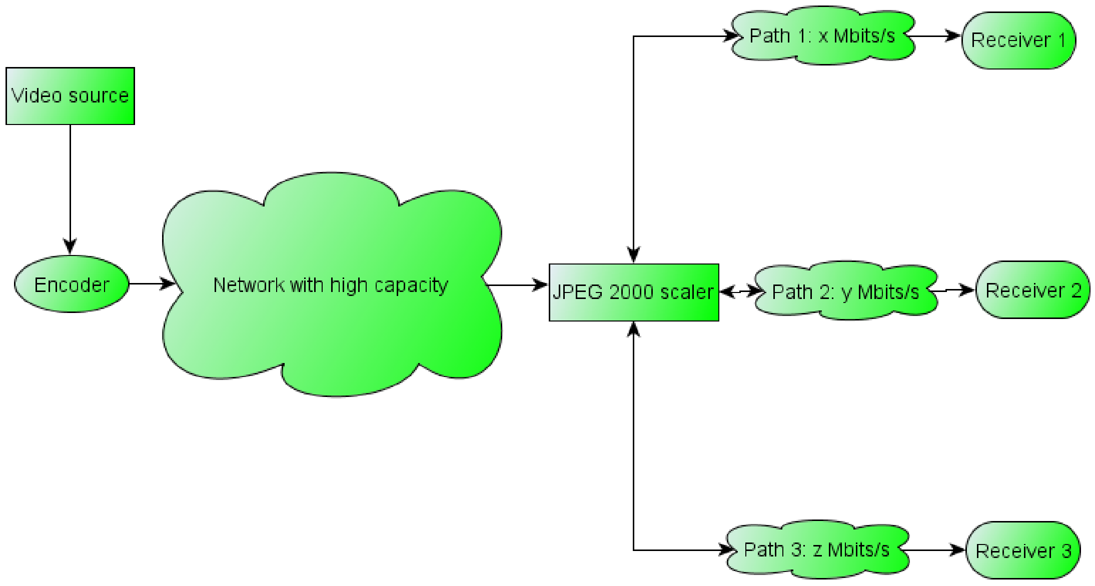


Figure 4.1: Networking scenario using JPEG 2000 scalability.

could be produced that said something about the visibility of sudden drops in bitrate, that metric could be used in the scenario described above to make a smart rate control at the node containing the JPEG 2000 scaler. It would, for example, be possible to let the bitrate follow the sudden drops in capacity experienced in an IP network, as long as the drop in capacity was lower than what was perceptible to a viewer. When the capacity dropped below the point of perceptibility, a smoother rate control solution could take over, or even buffering if the application allowed for it. This would ensure maximal quality in both cases, since it would smooth out sudden drops in bitrate whenever they were visible, but allow them when they were not, therefore making the quality stay at the highest possible level. The thesis will not aim to design such a rate control, however, only provide grounds for further work in that area. This is partly because a newly designed rate control should also be put through a subjective testing procedure. A subjective test must first be run in order to investigate the effects of drops in bitrate, and running two subjective tests is too time-consuming for a project like this.

Chapter 5

Methodology

The methodology chapter describes how results relevant to the problem statement were found. It starts by looking at the implementation of a JPEG 2000 scaler. Then it is shown how the scaler was used to perform certain experiments, leading to a method for finding an optimal number of quality layers to use in a JPEG 2000 image. The last part of this chapter deals with the creation of a subjective test, were the aim was to measure the effects of sudden reductions in video bitrate.

5.1 JPEG 2000 Scaler

In order to investigate the scalability feature in JPEG 2000, it was necessary to find a software program that would make it possible to extract quality layers from a JPEG 2000 codestream. This program should make it possible to specify a total output image size, which would be obtained by extracting a certain number of layers. In addition to this, the output image should be of the same format as the input, namely the JPEG 2000 codestream format. There currently exist two open source implementations of the JPEG 2000 codec in the programming language C. These are called OpenJpeg [9] and JasPer [8]. JasPer is also part of the JPEG 2000 standard as the reference implementation. Both of these codecs allows for the generation of quality layers, but only OpenJpeg provides a way of extracting the layers during decoding. However, only the number of layers can be specified, not the target image size. Since neither of the codecs had the necessary functionality, combined with the fact that a modification almost certainly would yield a program with lots of unnecessary bloat, it was decided to make a JPEG 2000 scaler from scratch.

A description of how the scaler was used in practice is provided in chapter 5.2. That chapter will, among other things, present a way of selecting an optimal number of quality layers. In the current chapter, the making of the scaler itself will be described.

5.1.1 JPEG 2000 Scaler Design Goals

The JPEG 2000 scaler was implemented with a few design criteria in mind. It needed to be fast in order to make it usable in a realtime scenario. It needed to be flexible, such that it could be extended with features needed in a real world scenario. Last, it had to be possible to write the program by one person in a fairly short period of time. By upholding these criteria, the resulting program would represent a prototype that could be used in the real world, and at the same time be completed in a short period of time.

To fulfil the last two criteria, it was decided to implement the most basic features first, and to make the code extendible by leaving certain functions declared, but without implementation. As an example, the scaler only works with the LRCP progression order, but by filling in content in an existing function, any other progression order can be supported. Equally, every marker segment in the JPEG 2000 codestream has predefined classes for storing information, and functions for reading in the information from the codestream. All that is needed is to fill in the read functions.

The criteria that the scaler should be fast was fulfilled by mainly two decisions. First, it was decided to write the scaler in the C++ programming language. C++ is a fairly low level language, and the compiled executables are often very efficient. Second, it was decided to add the start-of-packet marker to the JPEG 2000 bitstream, such that packet headers didn't need to be decoded. This decision was inspired by the implementation of the JPEG 2000 scaler described in chapter 3, where the authors state that "deep parsing" may be too slow for a realtime application [7]. This is of course highly dependent on the type of hardware used, so it is not a valid argument in all cases. More importantly, though, the usage of SOP markers made the implementation simpler, and quicker to implement. In future work, it could be interesting to do an implementation without SOP markers, and see if better results could be achieved. This might be possible due to the 6 bytes of space a SOP marker require.

5.1.2 JPEG 2000 Scaler Algorithm

In this section, the algorithm employed by the scaler will be described. The JPEG 2000 scaler works by discarding quality layers from an already encoded JPEG 2000 codestream (e.g. a .jpc image file encoded by the JasPer codec). It is important to note that the scaler does not encode or decode the image in any way, it exists purely as a middleman between the encoder and the decoder.

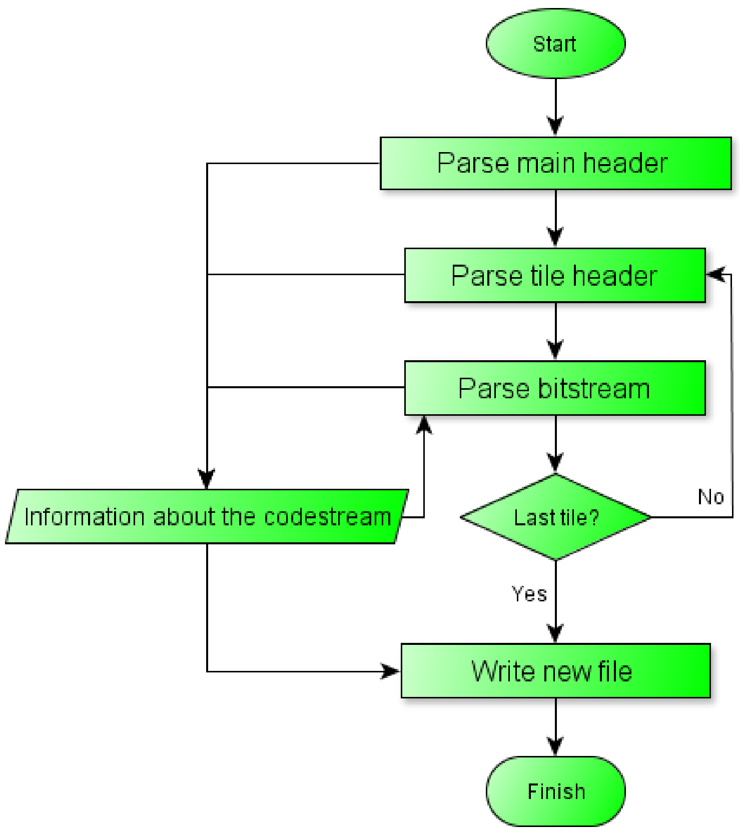


Figure 5.1: Flowchart describing the JPEG 2000 scaler.

An overview of the algorithm can be seen in figure 5.1. The algorithm works as follows:

Start

The program starts off by reading the entire image into a buffer. For all later processing, it is this buffer that is accessed. In a streaming application, such a procedure would introduce a delay of one frame plus the time it takes to process the image. It is still the favoured method when dealing with files on a hard drive, though, since reading in bytes one at a time from a file is a very slow procedure.

Before moving on to parsing the main header, the image is scanned for the start-of-codestream (SOC) marker. All JPEG 2000 codestreams starts with this marker.

Parse main header

The main header is parsed for information about the image, such as image size, number of components (colors), number of tiles, number of quality layers, and much more. The information is stored for later use.

Parse tile header

The tile header for the current tile is parsed. The tile header contains information about the bitstream, such as its size. It may also contain the same type of information as the main header. In such a case it overrides the information from the main header. This makes it possible for a tile to be encoded with separate parameter sets. The main header acts as the default in case the tile header does not contain any new information. Any new information in the tile header is stored for later used.

Parse bitstream

For each tile header, there is a corresponding bitstream. To parse the bitstream, information taken from the main and tile header is used. An example is the progression order used. In the scaler, only the LRCP progression order is supported. With this progression order, the lower layers come first in the bitstream. By using four for-loops, each value of L (layers), R (resolutions), C (components), and P (positions) may be looped through. For all such combinations of L, R, C, and P, a JPEG 2000 packet header must be decoded. However, in the current implementation, the decoding of the packet headers is bypassed by using the start-of-packet (SOP) marker. The bitstream is scanned until the next SOP marker occurs, thereby traversing the entire contents of a packet without decoding it. The bitstream position of each layer is found by recording the start and end positions of each layer loop. Pseudocode for this parsing procedure is provided below.

Write new file

When all tiles have been parsed, the output file is written. The positions of each layer in the bitstream is scanned (as recorded by the bitstream parsing), and by subtracting the start position from the end position, the size of the layer in bytes is acquired. The maximum number of layers possible without going over the target image size is then calculated, taking into account the size of the main and tile headers. The main and tile headers in the image buffer is updated with the new

Algorithm 2 Pseudocode describing the parsing of the bitstream

```
for all layers do
    record start of layer position
    for all resolution levels do
        for all components do
            for all positions do
                scan bitstream for the next SOP marker
            end for
        end for
    end for
    record end of layer position
end for
```

information, such as the new number of layers, and the new sizes of the bitstreams. Finally the buffer is modified such that layers higher than specified is discarded. The buffer is then written to the output file.

5.2 Experiments conducted with the JPEG 2000 scaler

Making the JPEG 2000 scaler made it possible to learn more about the limitations of the scalability feature, and to make sure that it is used to its fullest potential. The following sections describes the experiments that was conducted with the JPEG 2000 scaler, and what could be learned from them. In the results (chapter 6) later in this report, the results from these experiments will be presented.

5.2.1 Choosing the number of quality layers

In order to choose the right number of quality layers, it is first necessary to understand the limitations of scaled compression. After all, why isn't it possible to specify a huge number of quality layers, and therefore make it possible to extract an almost arbitrary bitrate from an image? Doing this without any sort of cost would be ideal, since it would achieve the exact same quality as compressing the image directly down to the desired bitrate. Scalability doesn't come without a cost, however, which will be explained below.

Comparison of direct and scaled compression

Using scaled compression to compress down to any point in between two layers will always yield an image with the bitrate defined by the lowest of the two layers. Direct compression will therefore outperform scaled compression more and more the farther from a layer boundary the desired bitrate lies. This effect is illustrated on the left side of figure 5.2. When the bitrate becomes high enough to cross a layer boundary, the whole layer will be included in the resulting image, and the gain from direct compression will rapidly decrease.

As this process continues to higher and higher bitrates (and therefore including more and more layers), the overhead associated with the increased number of layers will increase. This overhead consists of an increased number of packet headers and start-of-packet (SOP) markers. Due to this overhead, direct compression will get a higher and higher gain over scaled compression as the number of layers increase. However, as a counter-effect, the higher bitrates will also make the perceptual gain from direct compression smaller and smaller (this is because it is more difficult to spot quality differences at low compression).

To further complicate things, a higher total number of layers makes the distance from layer to layer smaller (see right side of figure 5.2). The gain from direct compression will therefore not have the space to grow as large between two layers as it did with a smaller total number of layers.

It is important to note that, so far, it is assumed that that the quality layers are equally spaced in terms of bitrate. This will also be the assumption during the rest of the report. However, it is possible that an unequal spacing is a better option, such that all the peaks shown in figure 5.2 would be of equal magnitude. Doing it this way corresponds to an equal spacing in terms of quality. The reason why this is not done is because the equal spacing in terms of quality would be different depending on the content of the image. An approximation could be made, but it is uncertain if it would work, and it would most certainly be very difficult to implement.

Procedure for finding the optimal number of layers

Taking the previous considerations into account, there are at least two ways to choose an appropriate number of layers. Both are based on the fact that direct compression yields by default the best achievable quality. The first method is to look at different total numbers of layers inside the desired bitrate range, and choose the one where the **average** gains from direct compression is the smallest.

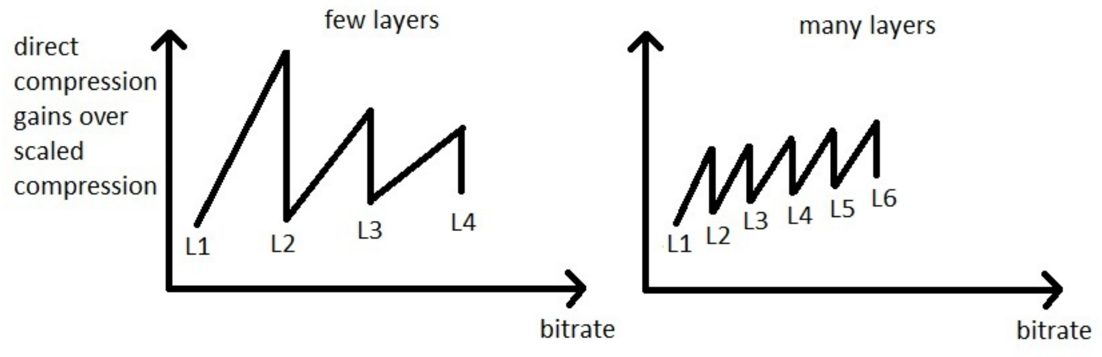


Figure 5.2: To the left: Few layers means that the distance between each layer is large, and that the direct compression reaches a large gain over scaled compression before the scaled compression can keep up. When the bitrate increases, the gains fast become less noticeable since the overhead is not great compared to the bitrate. To the right: The distance between two layers is small, and the gains from direct compression does not become very large in between two layers. As the bitrate increases, the overhead associated with the number of layers increase quickly, and direct compression gains more and more on the scaled compression.

Similarly, it is also possible to choose the one where the **maximum** gains from direct compression is the smallest. The latter is considered the safest choice, since it will make the direct compression gains differ less across the desired bitrate range. In this project, it was therefore decided to minimize the maximum gain from direct compression. The maximum gain can be observed in figure 5.2, just before each layer boundary.

The reason why it is possible to minimize the maximum gain from direct compression is because of the crossing point between increased overhead, and decreased distance between layers, as the number of layers increases. This point is illustrated in figure 5.3. The total gain from direct compression is the sum of these two contributions, and since they move in opposite directions, a minima is produced at a certain number of layers.

There is also another subtle point which has not been discussed yet, but has to be taken into account when looking for the optimal number of layers, and that is the increased encoding time required when producing more layers. The available software and hardware that is used for encoding, along with the requirements of the application, therefore defines an upper limit to the number of layers that can be used. This limitation is further discussed in chapter about realtime concerns 5.2.2.

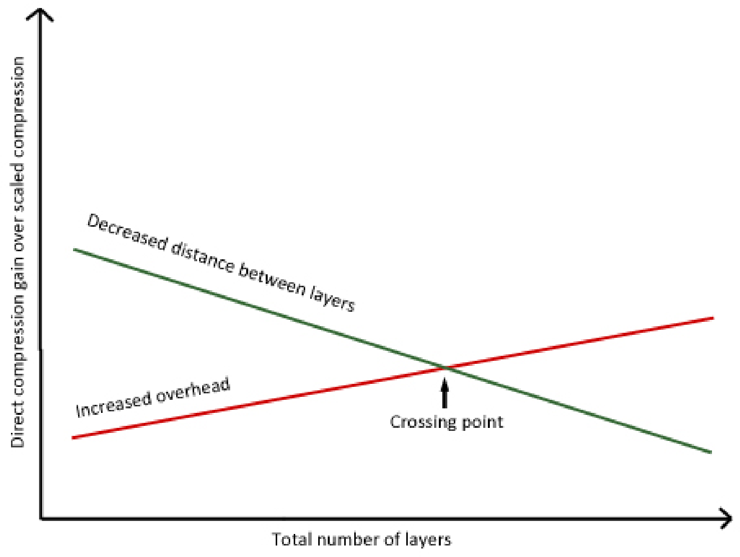

Direct compression gain over scaled compression
Decreased distance between layers
Increased overhead
Crossing point
Total number of layers

Figure 5.3: The crossing point between increased overhead, and decreased distance between layers, as the number of layers increases. The sum of these effects contribute to the gain in direct compression compared to scaled compression.

Practical implementation of the optimization procedure

In practice, the optimization procedure was done by comparing the SSIM values of direct and scaled compression on a representative set of images. Equation 2.3 was used for the comparison. SSIM was chosen over PSNR in order to better mimic the subjective quality difference. For more information about SSIM and PSNR, the reader is referred to the theory section of this report. The SSIM index was calculated by using a MATLAB script made by the author of the original SSIM paper [5]. Only default settings were used.

The minimization was done with a MATLAB script. The script generated different numbers of equally spaced quality layers for a set of 30 test images. From these images, the JPEG 2000 scaler was used to extract different bitrates. These bitrates were selected such that they were always one byte less than each of the layer boundaries, thereby forcing the scaler to throw away an entire layer each time. The resulting gain from direct compression over scaled compression, measured by equation 2.3, was recorded at the bitrate which generated the largest gain (averaged over the 30 test images). Put differently, the script found the largest peak in figure 5.2. This procedure produced a numerical value for each number of layers, describing the largest possible gain from direct compression for the bitrate range used. The number of layers to be used for this bitrate range could then be chosen according to which of them produced the lowest maximal gain from direct compression.

The test images used for the procedure were taken from the "Urban" section of the CSIQ database [10]. The reason why these images were chosen for the test was because they have successfully been used to test subjective image quality before [10]. In addition to this, the content in the images are quite varied, which makes the test applicable in more situations. Furthermore the images came in a relatively small resolution (512x512), which shortens processing times.

The optimization procedure was used to decide on the number of quality layers to use in the subjective test described in chapter 5.3. In that test, the resolution of the video was 1280x720 @ 25fps. To be able to use the previously described test images, the bitrate range used in the minimization procedure were modified so that they corresponded to the same amount of compression as the bitrates used for the subjective test. This was done very easily by modifying the bitrate by the following factor: $\frac{512x512}{1280x720}$.

5.2.2 Realtime concerns

When using the scalability feature in JPEG 2000 for video, there are two places where speed might be an issue. The first is at the encoder itself, and second at the extraction of quality layers. The extraction of quality layers is a relatively lightweight task, consisting of a one pass parse of the bitstream, and an update of the main and tile-part headers. In the case where one uses the start-of-packet marker for identifying the packets, the packet headers need not even be decoded. More concerning is the speed of the encoder itself. To achieve scalability, the encoder needs to decide on optimal truncation points for each quality layer. Moreover, for each new layer, a number of extra packets has to be assembled, and therefore extra packet headers needs to be encoded as well. The number of extra packets needed depends on the number of resolution levels, components, and precincts (see equation 5.1).

$$nPackets = nLayers * nResolutions * nComponents * nPrecincts \tag{5.1}$$

As mentioned in section 5.2.1, the number of quality layers will be bounded above by the encoding speed. For a given application, one therefore has to look at the expected number of resolution levels, components, and precincts, and at the available hardware/software, and of course the material that shall be compressed, to be able to find the upper limit to the number of quality layers.

To investigate the relationship between encoding speed and number of layers in practice, the image database described in section 5.2.1 was used. The average time it took to encode these images was recorded for different numbers of layers. A second test was also run, where four precincts were included. This made it possible to investigate the effect of equation 5.1. The results of this test can be found in chapter 6.2.

5.2.3 A special case: Lossless

For completeness, it is necessary to discuss one particular case where the procedures in this chapter falls short. Consider an image compressed with a rate of 0.9 (i.e. compressing down to 90 % of the size of the original). Depending on the complexity of the image, the JPEG 2000 codec is likely to compress the image even more efficiently than this, thereby reaching a near lossless compression (depending on the type of transform and quantization used). Another example is a one-color image, where the codec is likely to compress more efficiently than the specified rate. However, independently of the efficiency of the compression, the specified number of quality layers still has to be generated. In this case, the layers above the "lossless rate" consist of empty packets, signalled by a single bit in the packet header. Since the packet information is byte-aligned, a whole byte is still produced. In addition, the implementation used in this project uses the start-of-packet markers, consisting of 6 bytes each. The size of such an "empty" layer is therefore described in equation 5.2, given in bytes. A typical example with 5 resolutions, 3 components and 1 precinct gives 105 bytes for each empty layer.

$$szEmptyLayer = nResolutions * nComponents * nPrecincts * 7 \tag{5.2}$$

In most cases, this extra overhead is not important since the codec is already compressing way more efficiently than was what assumed when specifying the number and size of the layers. However, it does illustrate that one should not uncritically fill a JPEG 2000 codestream with quality layers, without considering the actual needs first.

5.3 Subjective test

The following chapters will describe how and why a subjective test was run.

As mentioned in the problem statement, one of the goals of this project was to examine in closer detail how sudden reductions in video bitrate affects the viewer. This information could in turn be useful when designing rate control systems for use in networks. The effects of such drops in bitrate may or may not be difficult to capture with objective quality metrics such as PSNR and SSIM. Therefore, it was decided to run a subjective test. The results from this test will be compared to the objective metrics in chapter 6.3.

When viewing a video clip containing a sudden reduction in bitrate, the reduction is naturally evaluated as causing a certain level of annoyance to the viewer. The overall quality of the clip is of less interest since it will depend on more factors than just the drop in bitrate, such as the overall bitrate used. The aim of the subjective test was therefore to find the point where the viewer would find the reduction in bitrate noticeable or annoying.

Before working on the test, a research protocol was written, explaining the background for the test, the testing procedure, and the test parameters. This protocol was approved by my supervisor before the test was run. The final version of this protocol can be found in the appendix A.2.

5.3.1 Summary of the test

This chapter will summarize the subjective testing procedure. A more elaborate explanation can be found in the subsequent chapters.

The subjective test consisted of 45 blocks (the five first were for stabilization only), each consisting of a reference clip and a test clip in succession. The test clip contained the same bitrate as the reference, except for the middle of the clip, where it went through a sudden reduction. This was immediately followed by a linear increase, lasting two seconds, back up to the reference bitrate. Altogether seven different reductions in bitrate were tested, along with the reference itself. The participants in the test were asked to rate the reductions in bitrate on the impairment scale shown in table 5.1.

For the practical purposes of the test, the standardized testing procedures described in ITU-R BT.500 ”Methodology for the subjective assessment of the quality of television pictures” [11] was used. More specifically, the DSIS (The double-stimulus impairment scale) method was used. An extension to this standardized method, ITU-R BT.710 ”Subjective Assessment Methods for Image Quality in High-definition Television” [12], was also used were applicable.

Table 5.1: The impairment scale used in the DSIS method [11].

5	Imperceptible
4	Perceptible, but not annoying
3	Slightly annoying
2	Annoying
1	Very annoying

5.3.2 Test set-up

The following sections will explain how the test was set up. The actual software used to generate the test is described in chapter 5.3.3.

The test room and equipment

The following settings were chosen in order to match the laboratory environment described in ITU-R BT.500 [11] and BT.710 [12] as close as possible.

The test room used was "Café Media" at NTNU. Inside this room, a testing area was set up, surrounded by thick blue curtains, such that almost no light entered. The TV used for the test was a 50 inch plasma screen (Samsung PS50C687). All signal processing in the TV was turned off during the test. The brightness and contrast of the TV was set up using PLUGE for HDTV [13]. This resulted in a screen luminance of 70 cd/m^2 when displaying peak white (measured with a Konica Minolta LS-100 Luminance meter). A light bulb was set up behind the TV with a luminance of approximately 0.15 of the peak screen luminance. This could not be accurately measured, however, due to lack of equipment. The chromaticity of the light was not measured. To display the testing sequence, the TV was hooked up to a computer outside the testing area via HDMI. The screen resolution set in the computer matched the resolution of the test sequence. The videolan media player [17] was used for playback. Three chairs were placed 180 cm from the TV screen, corresponding to a viewing distance of three times the height of the screen, as recommended in ITU-R BT.710 [12]. Viewers sitting in any of the three chairs were within +/-30 degrees horizontally from the center of the display. See figure 5.4 for a picture of the testing area. More pictures of the testing area can be found in the appendix A.3.

Participants

A total of 21 people participated in the subjective test, well over the recommended minimum of 15 participants [11]. Of these, six were women. All of the participants were university students in their twenties. Six wore glasses or lenses, but none of them had any other particular problems with vision (although this was not

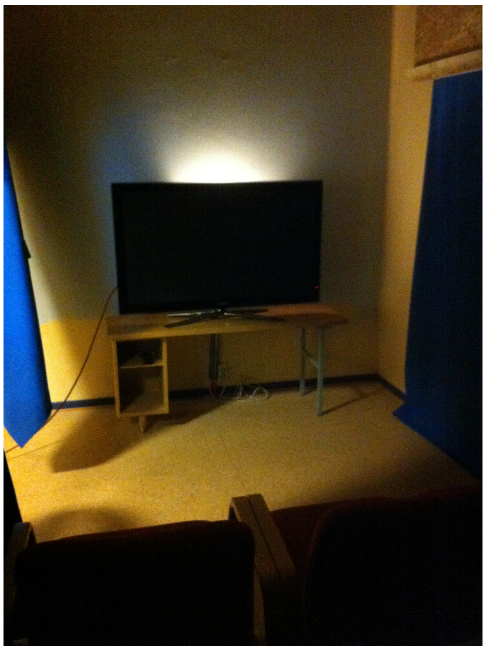


Figure 5.4: Picture taken inside the test room from behind the chairs. The curtains on the left side was closed during the test.

measured). All participants received both written and oral instructions about the testing procedure, and any questions the participants had was answered prior to the test. The written instructions can be found in the appendix A.4. The participants gave their votes by filling out a voting paper during the test. This paper can also be found in the appendix A.5.

Source material

The SVT high definition multi format test set was chosen as source material for the subjective test [14]. This is a set of five video clips, CrowdRun, ParkJoy, DucksTakeOff, IntoTree, OldTownCross, all 10 seconds long. They are described as demanding, but not unduly so. CrowdRun, ParkJoy, and DucksTakeOff are described as having a difficult coding difficulty. IntoTree and OldTownCross are described as having an easy coding difficulty. The set was chosen because the material has been assembled for the purpose of video quality testing (and therefore is of high quality), and because the content represents a relatively large range of test situations for such a small duration. The ParkJoy sequence, for example, is a sequence with fast movement, the CrowdRun is a sequence with a lot of movement and a lot of detail, while the OldTownCross and DucksTakeOff sequences are slower and less detailed. The fact that the sequences are so different will reflect in the subjective test results.

A snapshot of the parkjoy sequence (both reference and impaired) can be seen in figure 5.5.

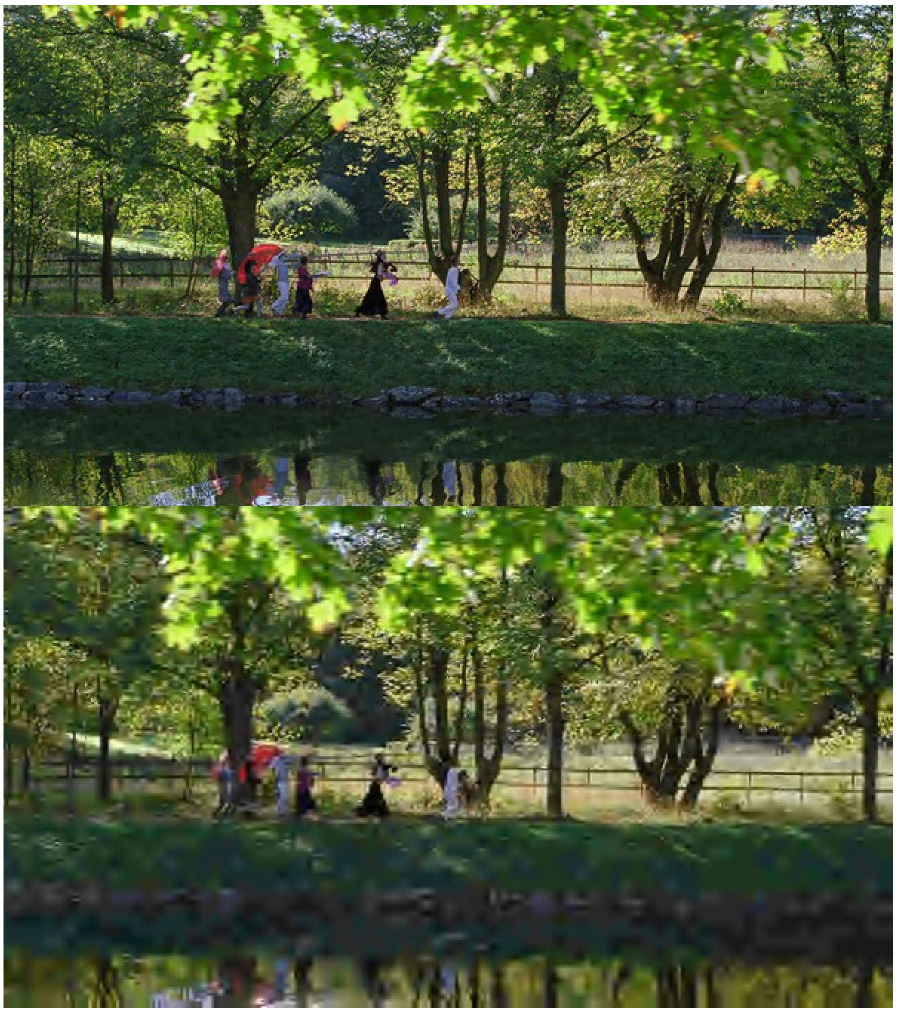


Figure 5.5: Snapshots from the parkjoy sequence. On the top is the reference clip, and on the bottom the lowest drop in bitrate is shown.

Preprocessing of the source material

For the test, it was chosen to use a resolution of 1280x720, with a framerate of 25 frames per second. This choice was made in order to have the largest possible quality, while still being able to play back the raw video without stuttering. The source material was acquired as 500 still frames per clip (corresponding to a framerate of 50 frames per second), each in 16 bit sgi image format. Every other frame was dropped such that the framerate equalled 25 frames per second. The images were to be encoded by the JasPer JPEG 2000 encoder [8], which does not handle sgi images. All the frames were therefore converted to the bitmap image format (bmp), using the ImageMagick image coding software [16]. This means that the images went from a 16 bit precision to 8 bit precision.

Test bitrates

It is recommended in ITU-R BT.500 [11] that the test itself doesn't last for more than 30 minutes, including any explanations and preliminaries. To stay within this limit, it was decided to use seven different bitrates for the test, along with the reference bitrate itself. The bitrates chosen were 2, 4, 6, 10, 15, 20, 25, and 30 Mbits per second, the last one being the reference itself. In a test clip, the bitrate would drop from the reference bitrate to one of these bitrates, and then increase linearly back up to the reference bitrate over a time period of two seconds. This procedure always happened in the middle of the test clip (from frame 100 to 149, where there were 251 frames in total). These drops in bitrate were chosen in order to span out the five point impairment scale as much as possible (see table 5.1), though this decision was only based on the subjective opinion of the author himself.

The duration of the linear increase back up to the reference bitrate was set to two seconds because of results presented in the master thesis currently being written by S. Tokheim and M. Markman [2], where it is found that most capacity drops in the public internet doesn't last for longer than about 200 ms. The time interval of two seconds used in this project is a good amount longer than this. However, because the bitrate is increasing linearly immediately after the drop, the time spent at the lowest bitrate is significantly less than two seconds (see figure 5.6). The two second interval allows for a worse behaviour than in [2], and at the same time a smooth increase back up to the reference rate. This will ensure that it is the drop itself that will be deemed annoying, not the increase.

The reference bitrate of 30 Mbits per second was chosen mainly because it represents the minimum quality needed for acceptable viewing with the particular resolution and bitrate used (this is the opinion of the author). It also represents a bitrate that is low enough to be streamed over e.g. the Internet (more information about using the public Internet for broadcasting can be found in [2]).

The lowest drop in bitrate can be observed in the snapshot of the parkjoy sequence in figure 5.5. This is a scaled down version of the snapshot, so the actual quality will of course seem better than it actually is.

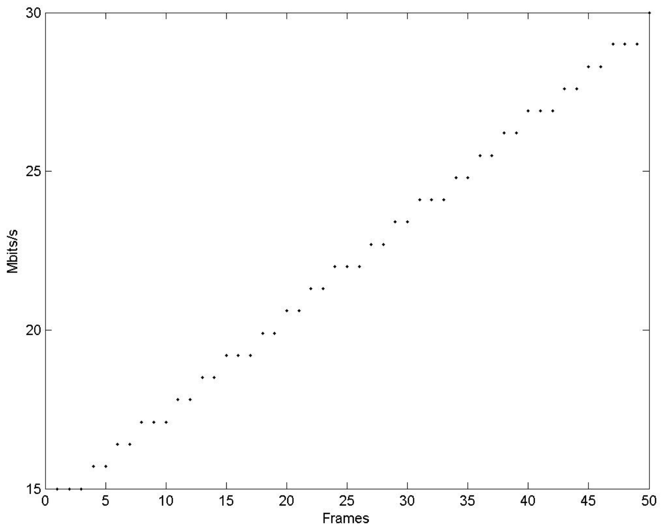


Figure 5.6: The actual bitrates produced when increasing the bitrates over two seconds (corresponding to 50 frames at 25 fps). This example shows the increase from 15 to 30 Mbits/s. The bitrate is increased in discrete steps defined by the quality layers. The bitrate is at its lowest for the duration of only 3 frames, which correspond to 120 ms.

Test timing

As recommended in the DSIS method [11], each testing condition was preceded by the unimpaired clip (reference) for 10 seconds, plus a mid-gray-level image for 3 seconds. The test condition (a clip containing a drop in bitrate), was then displayed for 10 seconds, followed by 7 seconds of a mid-grey-level image containing a number describing the number of the test condition. This number could then be compared to the number on the voting paper.

Overview of the test timing:

- 10s reference clip
- 3s mid-grey-level image
- 10s test condition (the video clip containing the drop in bitrate)
- 7s mid-grey-level image containing the test condition number

Test duration

Using the 8 different bitrates on the 5 different source clips gave a total of 40 clips to be shown in the test. The order of these were randomized. Furthermore, it was ensured that no two pairs of clips used the same source clip. This is in accordance with ITU-R BT.500 [11]. In addition to the 40 clips in the test, five more clips were added in the start of the test as a stabilizer. The result from these five clips were discarded. All together, the test had a duration of 22 minutes and 33 seconds (45 test blocks of 30 seconds each gives a total duration of 22 minutes and 30 seconds. The 3 extra seconds was because each source clip had 251 frames instead of 250, so the duration of a source clip was actually 10.04 seconds instead of 10 seconds).

It should be noted that only one test sequence was made, and so each participant viewed the same test. Ideally, the order should have been randomized for each participant. This was not possible due to the amount of time it took to generate the test.

5.3.3 Description of the software used to generate the test

The program used for generating the subjective test was implemented in MATLAB. To call the JPEG 2000 scaler implemented in this project, and the JasPer JPEG 2000 codec [8], the MATLAB function "system" was used.

First of all, the source sequences were encoded into the JPEG 2000 codestream format (jpc) [1]. The JasPer codec [8] was used for this purpose, using the following command line (simplified):

```
jasper --input i.bmp --output o.jpc -O rate=m -O sop -O prg=lrcp
       -O numrlvls=5 -O ilyrrates=r
```

This command line specifies an input and output image, a max rate (m), the usage of the start of packet marker (sop), the usage of the lrcp progression order, the number of resolution levels (5), and intermediate rates less than the max rate (r). The max rate, m, was set to correspond to 30 Mbits per second, while the intermediate rates, r, was a comma-separated list of 39 rates equally spaced from 2 Mbits per second up to, but not including, 30 Mbits per second. This means that, in total, there were 40 quality layers embedded into the bitstream. This command was performed on each of the images in the source clips.

After encoding, the resulting images were passed through the JPEG 2000 scaler with the following command line (simplified):

```
Jp2parser -i i.jpc -o o.jpc -b r
```

This command line specifies an input and output image, and a target size in bytes. The output image is as close to the target size as possible without going over. This command was performed on all encoded images with different target sizes in order to achieve the test bitrates discussed earlier. Here it must be noted that the test bitrates could not be achieved exactly, since only an integer number of quality layers may be used. The same goes for the linear increase in bitrate back up to the reference bitrate, which happens in steps defined by the quality layers (see also figure 5.6).

In addition to this, mid-grey-level images was also generated. Some of these images required numbering as well. This was done with the following command line (simplified):

```
convert i.bmp -font Arial -pointsize 150 -gravity Center
        -fill white -annotate 0 n o.jpc
```

This command line puts a white coloured number, n, in the middle of the input picture. In this case, the input picture was a mid-level grey image. The convert program is included in the free ImageMagick image software [16].

All the generated images were then numbered from 1 and up, and put in a separate folder. This made it possible to assemble the entire sequence of images using ffmpeg [15]. The following command line was used to achieve this:

```
ffmpeg -f image2 -r 25 -i Temp/%%05d.jpc -pix_fmt yuv420p
       -vcodec rawvideo -an out.avi
```

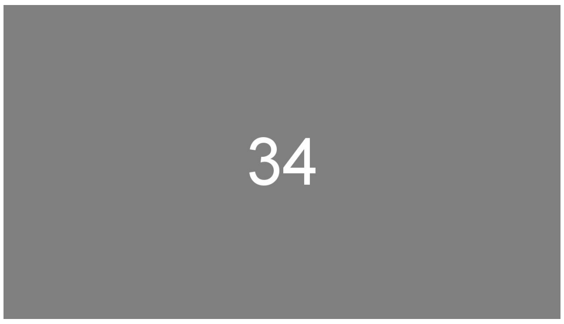


Figure 5.7: Grey screen with number on it, as shown at the end of each test sequence.

This command line specifies that images from the folder Temp numbered from 1 and up, with up to 5 preceding zeros, having a framerate of 25 frames per second, should be raw-encoded with the yuv420p pixel format, and that no sound is included. The yuv420p pixel format means that the chroma components of the color space have been subsampled by a factor of two. This choice was made in order to get correct colors during playback in the Videolan media player [17].

5.3.4 Analysis of the test results

In order to analyse the results from the test, the method described in ITU-R BT.500 [11] was used. First, the mean scores over all the observers was calculated, yielding altogether 40 numbers, one for each combination of test bitrate and source clip. The 95 % confidence interval was calculated for each of the values, using equation 5.3.

$$\delta_{sb} = 1.96 \frac{S_{sb}}{\sqrt{N}} \tag{5.3}$$

Where N was the total number of observers, s is the subscript for source clip, and b is the subscript for bitrate. S_{sb} is the standard deviation, and can be calculated by equation 5.4.

$$S_{sb} = \sqrt{\sum_{i=1}^{N} \frac{(mean_{sb} - vote_{sbi})^2}{N-1}} \tag{5.4}$$

The calculated confidence intervals are valid as long as the distribution of votes are normal. The β_2 test was run to check for this.

$$\beta_{2sb} = \frac{m_4}{m_2^2} \tag{5.5}$$

where

$$m_x = \frac{\sum_{i=1}^{N} (vote_{sbi} - mean_{sb})^x}{N} \tag{5.6}$$

If β was between 2 and 4, the distribution was assumed to be normal.

ITU-R BT.500 also recommends the removal of outliers, but only if the number of observers are less than 20. In the test described here, 21 observers participated, and so no outliers were removed.

5.3.5 Sources for error

Because a subjective test is dependent on human observers, there are several sources of error that could affect the results. Things such as light entering the test area, and noise from the outside hallway, could both provide a source of distraction. To remedy this, the test area was made sure to be properly closed during the test session. The noise from the hallway could not be controlled, but during the test sessions it was noted that the noise was minimal.

Another source of distraction was the other observers, as there were up to three observers per test session. They were all told to be silent during the test, however, which to the authors knowledge was uphold.

In the test set up, the background luminance and chromaticity could not be measured. It is believed that this would be of greater importance if the results were to be compared with another laboratory with the same set up, but in this case the results stand on their own, and it is believed that they are not greatly affected.

In addition to the environmental effects discussed so far, the test itself could be a source of errors. The most obvious things that could be improved were the number of observers, source clips, and the number of test bitrates. Increasing these parameters would give more fine grained and concentrated results. There is always this compromise between time and resources required, and the size of the test.

A less obvious thing that could affect the results was the way the bitrates were increased after the drop. A linear increase over two seconds means that the quality quickly returns to a higher level. A moment of not paying attention could make the observer not notice the drop at all. The observers were informed about the position of the drops in bitrate to help minimize this effect. In addition, it is highly unlikely that more than a few of the 21 observers would let the drops in bitrate go unnoticed per test.

The goal of the subjective test was to investigate what it takes in terms of drops in quality or bitrate before an observer notices it or finds it annoying. It is problematic to measure this in only 10 second tests, since a drop in bitrate might be more annoying if it occurs often, than if it occurs, say, only every 30 minutes. It might also be dependent on the type of content shown, and on which device it is viewed. Another problem is that the reference bitrates were fixed at 30 Mbits/s. At this bitrate, the test clips were already noticeable impaired. This means that the noise induced by making the bitrate drop could be masked by the noise that was already there. These types of effects can only be accounted for by doing more tests, and especially by testing a longer test clip simulating a real world viewing situation. The results from this project should therefore not be extended too far beyond the context in which the test was run, and should be used with care.

Chapter 6

Results And Discussion

6.1 Number of layers

In this chapter, results from the number-of-layers optimization procedure is presented.

In figure 6.1, the results are shown for the bitrate range used in the subjective test (2 to 30 Mbits/s). The y axis in the plot shows the maximum gain in direct compression over scaled compression, measured with equation 2.3 (the equation used for comparing two SSIM values). This means that, if the SSIM index was a perfect subjective quality metric, the y axis would show the subjective gain factor in favour of direct compression. The higher the value, the worse the scaled image would perform in comparison to a directly compressed image.

It is seen that when using a total of 40 equally spaced quality layers, direct compression performs approximately 18 percent better than the scaled compression in the worst case. This is the best worst-case behaviour for this bitrate range.

Since 40 quality layers represents the lowest maximal gain from direct compression, it was used in the subjective test. According to figure 6.1, though, it seems that the loss by increasing the number of layers from 40 and up is not that great, so why stop at 40 layers? It was decided that 40 layers already gives a fairly fine grained control of the bitrate (a granularity of $\frac{30-2}{40} = 0.7Mbits/s$), and that it would be unnecessary to accept any more losses, however small. The encoding time was also kept shorter by staying at 40 layers. In addition to this, the equation used to compare SSIM values (equation 2.3) is not very well tested. The actual values produced must therefore not be given too much attention. The fact that there is a minimum value is still valid, though, as explained in chapter 2.1.2.

At around 10 layers, there seems to be a spike in the gain factor. The reason why such spikes might occur is because the layer boundaries are placed on different bitrates depending on the number of layers used. Because of these differing positions, the gain from direct compression over scaled compression will also differ. Regardless of the reason for the spike, though, the subjective test used a number of layers that were far from the spiked area. The results from the subjective test should therefore not be affected.

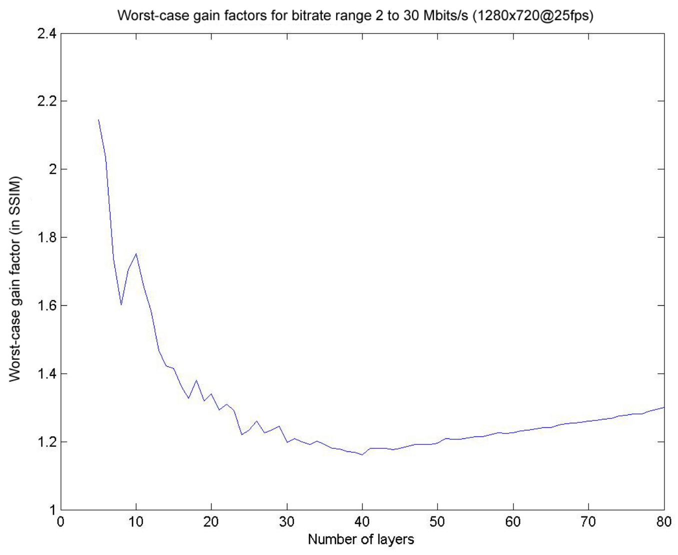


Figure 6.1: Maximum direct compression gains over scaled compression for different total numbers of equally spaced layers. The bitrates (2-30 Mbits/s) are for a resolution of 1280x720 at 25 frames per second.

As a sanity check, it is worth looking into the different number of layers, and see what the actual gain factors are for each of the bitrates in the range, not just the worst-case one. In figure 6.2, 6.3, and 6.4, a total of 10, 40, and 80 quality layers are used, all inside the range of 2 to 30 Mbits/s. A total of 100 equally spaced bitrates in the range were used in order to generate the graph. Some deviation from the actual values will therefore occur. It is important to keep in mind that the graphs does not have the same limits on the y-axes. This was done in order to make it easier to examine the details of the curves.

As expected, a low total number of layers (figure 6.2) yields large distances between layers, and the gain factor will grow large in between. As the total number of layers increases (figure 6.3 and 6.4), the distance between layers decreases, but the overhead associated with the increased number of layers increases. The worst-case behaviour is therefore minimized somewhere in the middle. In this case, it is minimized at 40 layers.

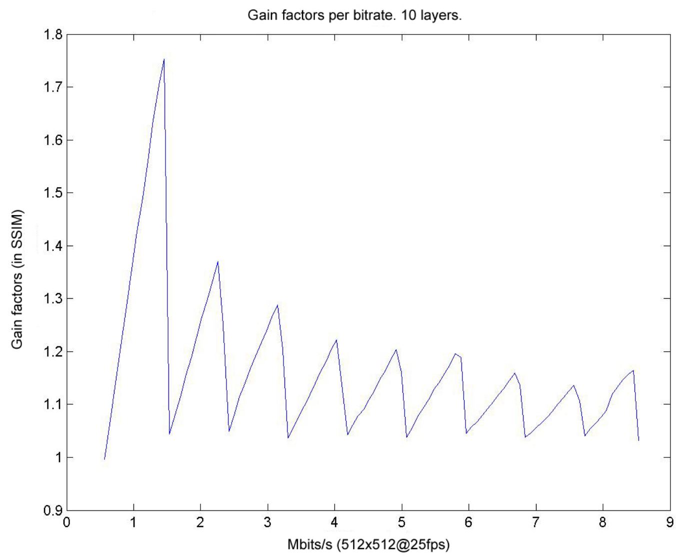


Figure 6.2: The direct compression gain factors (measured in SSIM) for each bitrate in the range. 10 quality layers are used.

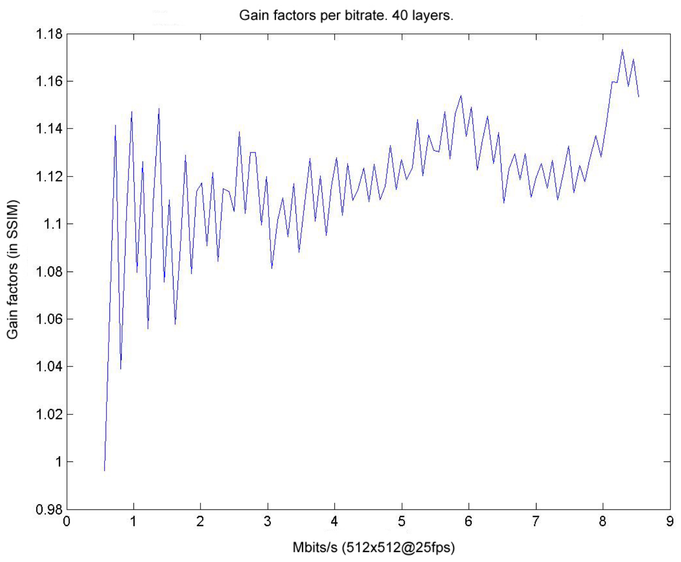


Figure 6.3: The direct compression gain factors (measured in SSIM) for each bitrate in the range. 40 quality layers are used.

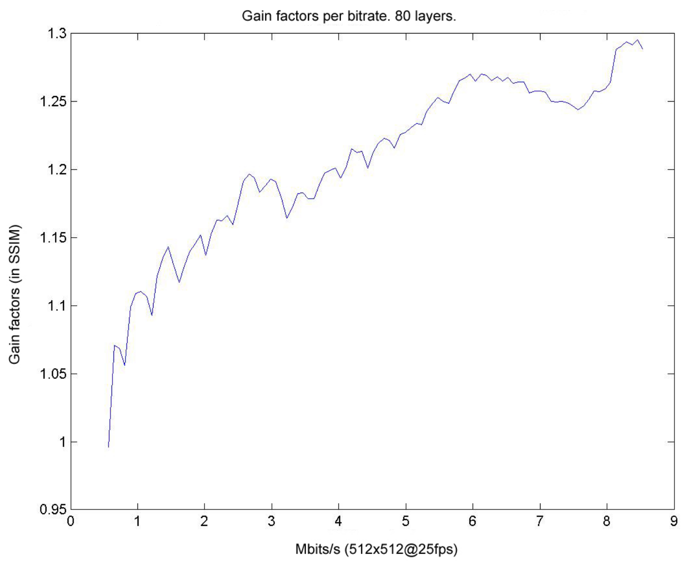


Figure 6.4: The direct compression gain factors (measured in SSIM) for each bitrate in the range. 80 quality layers are used.

In figure 6.5, the results from the optimization procedure are shown for another bitrate range. This time the bitrates correspond to 10 Mbits/s and 30 Mbits/s, at a resolution of 1920x1080 and 24 frames per second. Here the result is quite different from before. The best worst-case behaviour is found at around 15 layers, and here the direct compression is only about 12 percent better than scaled compression. This illustrates that the optimal number of layers is highly dependent on the bitrate range (or more accurately, the compression range).

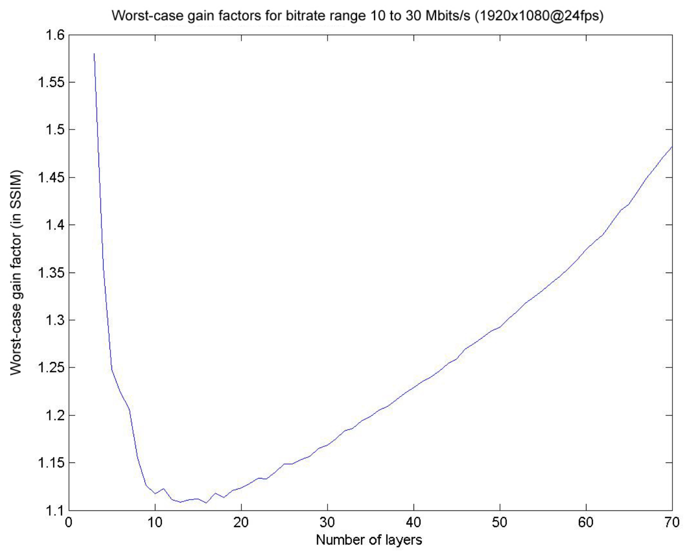


Figure 6.5: Maximum direct compression gains over scaled compression for different total numbers of equally spaced layers. The bitrates (10-30 Mbits/s) are for a resolution of 1920x1080 at 24 frames per second.

6.2 Speed test

The results from the speed test is shown in figure 6.6. As expected, the encoding time increases as the numbers of layers to encode increases. This fact is magnified by using several precincts as well. This is essentially a confirmation that it is not only the truncation point decision-making that takes time, but also the encoding of packet headers, and placing the packets in the bitstream (since more precincts require more packets; see equation 5.1). The actual number of seconds taken to encode is very dependent on the available hardware, and the implementation of the software, so the number of seconds presented here is only relative to those dependencies. The most important thing to take away from the curve is that the number of layers to use is dependent on the application. If the application requires a large amount of precincts or other progression types, the encoding time will increase faster as the number of layers increase. This puts a cap on how many layers one can use, independent of the optimization procedure described earlier.

As for the figure itself, it is not a perfectly straight line as might have been expected. This is because the test conditions couldn't be controlled very accurately. For example, the JasPer JPEG 2000 encoder executable was called by the MATLAB "system" function. This function is responsible for setting up a new process for the executable. The time this takes is dependent on the current situation of the operating system, thereby making it difficult to get consistent time measurements. The times presented here should therefore be taken only as a guideline to how encoding time evolves as the number of layers increases.

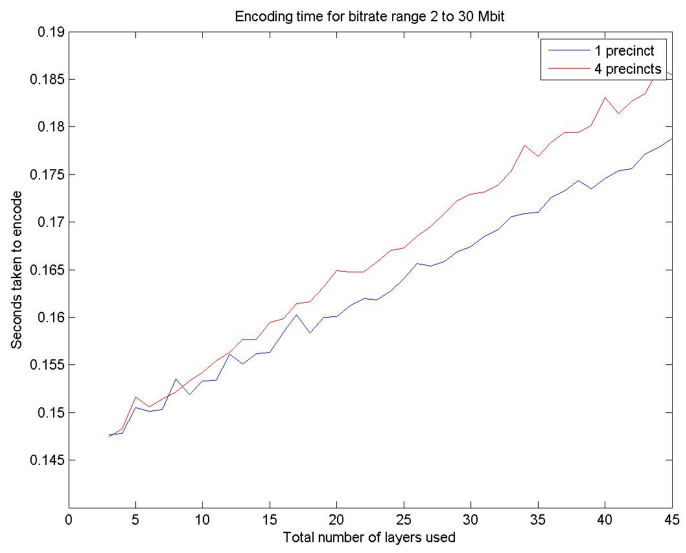


Figure 6.6: Encoding speeds for different numbers of layers. A second measurement using 4 precincts is also included.

6.3 Results from subjective test

In order to present the results from the subjective test in a reasonable amount of space, figures from only two of the source clips will be presented. The clips chosen are CrowdRun and OldTownCross. CrowdRun represents the worst subjective results, while OldTownCross represents the average behaviour. The figures for the three other source clips can be found in appendix A.1. Average results over all the source clips will not be shown. This is because the number of source clips is low, and the results would therefore deviate more as a result of the differences in the source clips, than the differences in the subjective evaluations.

Figure 6.7 and 6.8 show the mean score given by the observers, along with a line segment representing the 95% confidence interval. The scores are given on the impairment scale in table 5.1 (5 representing "imperceptible", and 1 representing "very annoying"). The bitrates in the figures are the target bitrates for the sudden drop that was described in the methodology (chapter 5.3). In table 6.1, the numerical mean values is shown for all the source clips and bitrates. For the OldTownCross clip, it is seen that the majority of the observers found the drop in bitrate imperceptible as long as the target bitrate was kept above 10 Mbits/s. At 10 Mbits/s, the majority of observers found the drop in bitrate somewhere between "Perceptible, but not annoying", and "Slightly annoying". For the CrowdRun clip, the point at which the drop in bitrate became "Perceptible, but not annoying" was approximately 15 Mbits/s.

Bitrate [Mbits/s]	ParkJoy	OldTownCross	IntoTre	DucksTakeOff	CrowdRun
2	1.7143	2.0476	2.0000	2.0952	1.5238
4	2.1905	2.5238	2.0476	2.4762	1.9524
6	2.6190	2.6190	2.6190	3.4762	2.0476
10	3.4762	3.6667	3.6667	4.4762	2.9524
15	4.4762	4.7619	4.5714	4.6667	3.6667
20	4.7143	4.8095	4.8095	4.9524	4.4762
25	4.7143	4.8095	4.8571	4.9048	4.8095
30	4.8095	4.7143	4.9524	4.8571	4.8095

Table 6.1: Mean score value for each source clip and test bitrate.

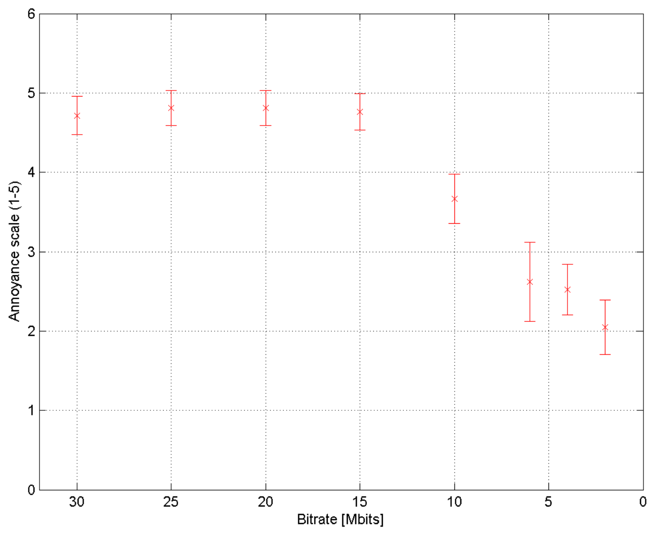


Figure 6.7: Mean grades per bitrate for the OldTownCross clip. The line segments marks the 95 % confidence intervals.

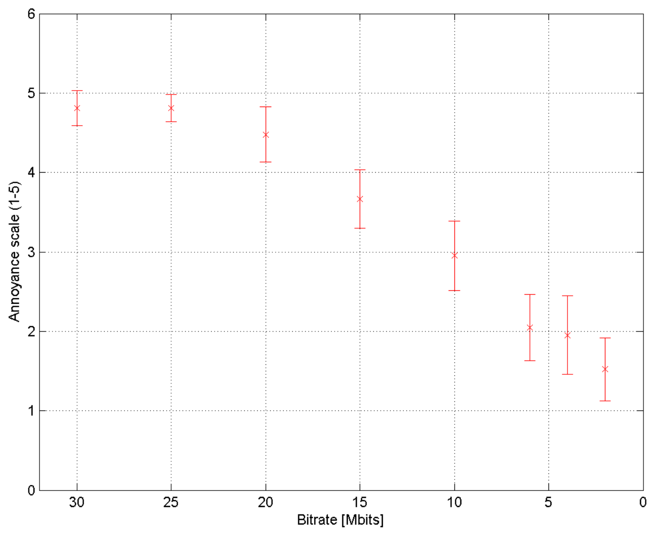


Figure 6.8: Mean grades per bitrate for the CrowdRun clip. The line segments marks the 95 % confidence intervals.

The confidence intervals shown are quite small relative to the grading scale, meaning that there is a relatively clearly marked area where the drop in bitrate becomes "Perceptible, but not annoying". In table 6.2 the β_2 values for each combination of bitrate and source clip is shown. A value between 2 and 4 indicates a normal distribution. It is seen that most of the values are in this range, but some of them deviate quite a bit. Figure 6.9 and 6.10 shows the distribution of the results for each bitrate for the the CrowdRun and OldTownCross sequences. Here it is seen that the results are quite concentrated at high bitrates. The fact that the results are so concentrated at some bitrates could also explain why they are not normally distributed.

Bitrate [Mbits/s]	ParkJoy	OldTownCross	IntoTre	DucksTakeOff	CrowdRun
2	4.5332	2.9846	2.2969	2.9854	3.7508
4	2.4687	2.6693	1.9468	2.7884	3.3791
6	2.1170	2.6193	1.9111	2.4091	1.7495
10	1.8996	2.8711	2.2232	1.0091	2.6765
15	2.6478	6.6591	5.1899	4.5355	2.1694
20	5.4774	8.8815	3.4853	19.050	2.4209
25	5.4774	8.8815	5.1667	8.6053	3.4853
30	3.4853	5.2004	19.050	5.1667	8.8815

Table 6.2: β_2 values for the subjective score values for the different combinations of bitrate and source clips. A value between 2 and 4 indicates a normal distribution.

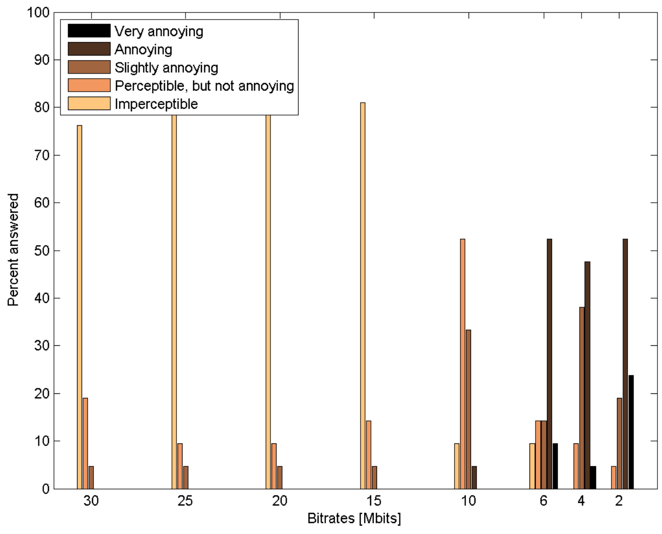


Figure 6.9: The distribution of votes for each bitrate for the OldTownCross clip.

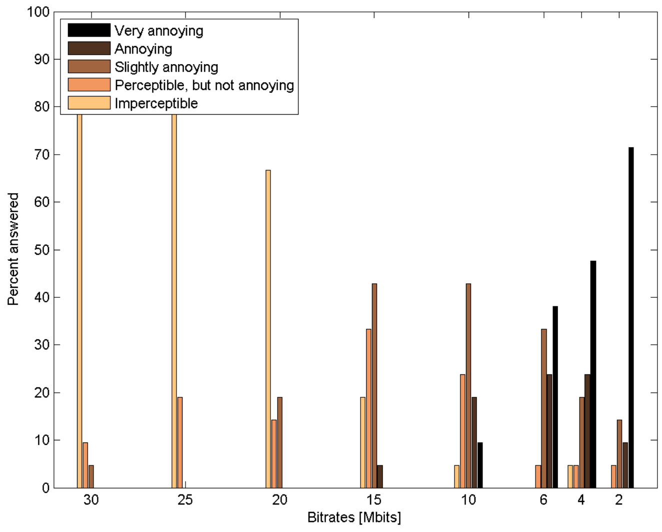


Figure 6.10: The distribution of votes for each bitrate for the CrowdRun clip.

In figure 6.11 and 6.12, the SSIM values for the two source clips are shown. More precisely, they are the SSIM values for the first frames that experienced a drop in bitrate. The values are calculated with respect to the reference bitrate (the reference, 30 Mbits/s, is therefore not shown in the graphs). The numerical values used for these graphs, along with the corresponding PSNR (Luma) values can be found in appendix A.1. All SSIM values are calculated with the MATLAB script made by the author of the original SSIM paper [5]. Only default parameters are used.

The whole source clips were also compressed down to each test bitrate, and the mean SSIM and PSNR (Luma) values was calculated over the entire source clips. Tables for these values can be found in appendix A.1. It turned out that these values were very similar to those measured only at the frame with the drop in bitrate, so they are not shown here (they were similar enough to not affect the discussion presented here). This was done to confirm that nothing special happened at the frame with the drop in bitrate.

As expected, the SSIM values steadily decrease as the bitrate decreases. In figure 6.7 and 6.8 it was observed that at the higher bitrates, the observers did not notice the drops in bitrate. This means that, when it occurs in small drops, it takes a certain amount of quality degradation for people to be able to notice it. In tables 6.3, 6.4, and 6.5, the bitrates, subjective scores, and SSIM and PSNR (Luma) values, are shown for three different areas, "invisible", "visible", and "annoying". "Invisible" is the area for which the confidence interval lies entirely above "Perceptible, but not annoying". "Visible" is the area which is not "Invisible", and the confidence interval lies entirely above "Slightly annoying". "Annoying" is the area which is not "visible" or "invisible". The table shows only the occurrence closest to "Perceptible, but not annoying" of each of these areas. By grouping the values in such a way, it is hoped that any differences caused by the content in the source clips themselves won't affect how the objective metrics perform.

It is interesting to see how much the values are still affected by the different source clips. In the ParkJoy sequence, the drop in bitrate is invisible all the way down to an SSIM value of 0.9265, while for OldTownCross, it has become annoying already at 0.9403. This is a strong indication that SSIM is not a good tool to measure how visible a sudden drop in bitrate is. As expected, PSNR fares no better, maybe even worse. It is seen that the PSNR value for the "Invisible" area varies by as much as almost 10 dB.

Clip name	Invisible			
	Mbits	Score	SSIM	PSNR
CrowdRun	20	4.4762	0.9747	31.7310
OldTownCross	15	4.7619	0.9759	36.2039
ParkJoy	15	4.4762	0.9265	28.3083
IntoTree	15	4.5714	0.9496	36.3536
DucksTakeOff	10	4.4762	0.9305	26.8412

Table 6.3: Table showing the bitrates, subjective scores, SSIM, and PSNR (Luma), in the invisible area.

Clip name	Visible			
	Mbits	Score	SSIM	PSNR
CrowdRun	15	3.6667	0.9526	29.2062
OldTownCross	10	3.6667	0.9615	32.7652
ParkJoy	10	3.6667	0.8957	25.4679
IntoTree	10	3.6667	0.9150	33.7381
DucksTakeOff	6	3.4762	0.9004	24.4569

Table 6.4: Table showing the bitrates, subjective scores, SSIM, and PSNR (Luma), in the visible area.

Clip name	Annoying			
	Mbits	Score	SSIM	PSNR
CrowdRun	10	2.9524	0.9159	26.2180
OldTownCross	6	2.6190	0.9403	30.1200
ParkJoy	6	2.6190	0.8362	23.2726
IntoTree	6	2.6190	0.8766	31.5183
DucksTakeOff	4	2.4762	0.8299	23.2291

Table 6.5: Table showing the bitrates, subjective scores, SSIM, and PSNR (Luma), in the annoying area.

It is especially interesting to note the results from the three test clips OldTownCross, ParkJoy, and IntoTree. According to the observers in the test, it took almost the same amount of drop in bitrate for them to notice it (the same holds for the visible and annoying areas). This is in disagreement with the objective quality measured in those same clips, as seen in table 6.3, where the SSIM values varies from 0.9265 to 0.9759. At the annoying area (table 6.5) the SSIM values vary even more, from 0.8362 to 0.9403. It seems, therefore, that it is not possible to use SSIM or PSNR (Luma) to predict when a drop in bitrate becomes noticeable. However, it is still clear from this test that the drop in bitrate has to have a certain magnitude for it to be visible to the viewer, in this test at least 10 Mbits/s as in the CrowdRun clip. The exact amount of drop in bitrate that is unnoticeable to the viewer seems more difficult to estimate, at least with SSIM or PSNR.

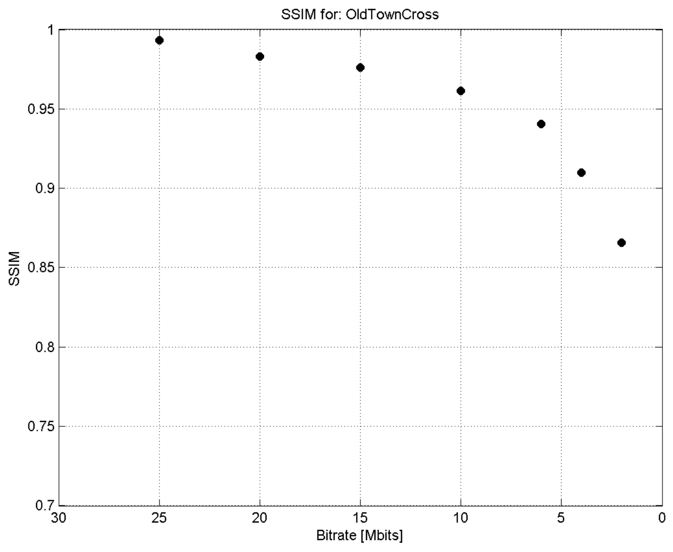


Figure 6.11: SSIM index with respect to the reference bitrate for the OldTownCross clip.

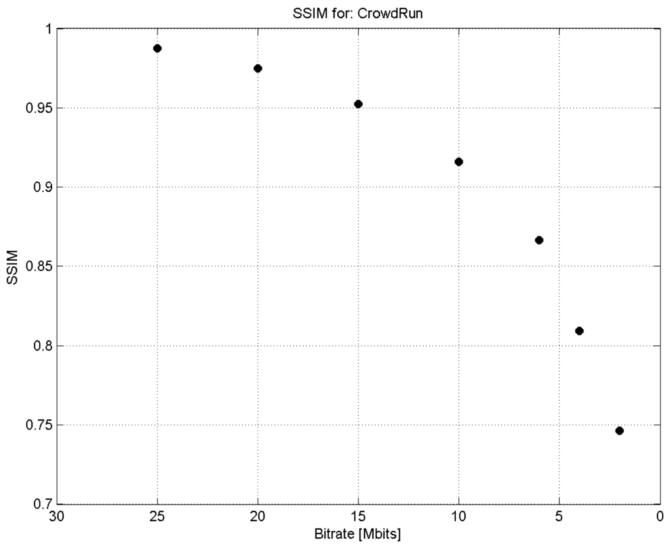


Figure 6.12: SSIM index with respect to the reference bitrate for the CrowdRun clip.

The results found from the subjective test were not as expected. It was expected that the objective quality metrics could, to a certain degree, tell how visible a drop in bitrate was. Since this was not the case, it is worth asking if there could have been any errors in the calculations. There is always the possibility of such errors, but the values presented so far behave in such a way that it is unlikely they are very far off. Both the SSIM and PSNR (Luma) values decrease as the bitrate decreases. The decrease is also monotonic, as it should be. The same goes for the results from the test itself. The degree of annoyance decreased in a way that was expected, and most of the observers agreed on the results, as evidenced by the highly concentrated scores. It is only when the objective metrics are compared to the subjective ones that the results differ from what was expected. A reason for this could be that the subjective scores are given on an annoyance scale rather than a quality scale, and that therefore the objective quality metrics measure something different. If this is indeed the case, it is still a surprise that the quality at the dropped bitrate has so little predictive capability with respect to the degree of annoyance. For future work, it would be very interesting to see if there are other parameters that are better tools for measuring the visibility of drops in bitrate. Possible parameters that could have an impact include the duration of the drop, the way the bitrate is increased back up again, and the quality at the reference bitrate.

Chapter 7

Conclusion

In this report, a method has been described that determines the optimal total number of quality layers to embed into a JPEG 2000 codestream. The method assumes equally spaced quality layers in terms of bitrate, and works by minimizing the maximal difference between direct and scaled compression. For a bitrate range of 2 to 30 Mbits/s and a video resolution of 1280x720 at 25 frames per second, the optimal number of layers is found to be 40. This result is found by using a newly implemented JPEG 2000 scaler. The scaler uses the start-of-packet marker in order to parse the codestream. It is shown that the optimal total number of quality layers is highly dependent on the desired bitrate range, and is capped above by the speed of the encoder. This is because a JPEG 2000 encoder spends more time embedding more quality layers in the stream. It is also shown how this fact is magnified when using several progression types at the same time, e.g. precincts.

The JPEG 2000 scaler was also used in the implementation of a subjective test. The test was observed by 21 participants, who was asked to rate sudden drops in bitrate on an annoyance scale. The reference bitrate was 30 Mbits/s, and the test bitrates at the drops were 2, 4, 6, 10, 15, 20, and 25 Mbits/s. After a drop, the bitrate was linearly increased back up to the reference bitrate over a time period of two seconds. It was found that the bitrate had to be dropped down to at least 20 Mbits/s before the drop became perceptible to the observers. In one case, the bitrate could be dropped all the way down to 10 Mbits/s before it became perceptible. SSIM and PSNR (Luma) values was calculated to see if there was any correlation between the visibility of the drop in bitrate and the predicted quality at the level of the drop. It was found that neither SSIM nor PSNR (Luma) could predict the visibility of the drop. As an example, in one test clip the drop became noticeable at an SSIM value of 0.9265, while another test was perceived as slightly annoying already at an SSIM value of 0.9403. It is concluded that, as expected, the bitrate will have to drop a certain amount before the drop becomes perceptible. However, the exact amount is difficult to estimate, at least with the metrics used in this project.

7.1 Future work

In the future, it can be interesting to see if an equally spaced number of quality layers in terms of quality (as opposed to bitrate) would yield an even better procedure for choosing the number of layers. This might prove challenging, since the spacing would be different depending on the content. In addition, a JPEG 2000 scaler could be made without the use of start-of-packet markers. This would be interesting because it would remove the extra space required by those markers (6 bytes per marker).

It would also be interesting to investigate if there exist better parameters for predicting the perception of drops in bitrate. Possible parameters include the duration of the drop in bitrate, the way the bitrate is increased back up to the reference, and the level of the reference bitrate itself.

When such parameters are found, a rate control can be designed which discriminates between visible and invisible drops in bitrate, and acts accordingly. This will be of interest in a networking scenario, where sudden drops in bitrate are common.

Bibliography

[1] ITU-T T.800 *JPEG 2000 Part 1: Core Coding System.* 2002

[2] Master thesis by Stian Tokheim and Martin Markman. At the time of writing, the thesis is not yet finished. *Reliable broadcast contribution over the public internet* Spring 2012

[3] Rong-Yu Qiao and Michael H. Lee *Adaptive Rate Control of Motion-JPEG2000 video over IP networks.* 2007

[4] G. Yang, L. Chen, T. Sun, M. Gerla and M. Y. Sanadidi *Smooth and Efficient Real-time Video Transport in Presence of Wireless Errors* 2006

[5] Zhou Wang, Alan Conrad Bovik, Hamid Rahim Sheikh, Eero P. Simoncelli *Image Quality Assessment: From Error Visibility to Structural Similarity.* 2004. Matlab code for calculating SSIM can be found at `https://ece.uwaterloo.ca/~z70wang/research/ssim/`.

[6] Zhou Wang, Alan Conrad Bovik *Mean Squared Error: Love It or Leave It?.* IEEE Signal Processing Magazine 2009

[7] Rong-Yu Qiao, Michael H. Lee and Keith Bengston *Motion-JPEG2000 Stream Scaling for Multi-Resolution Video Transmission* 2008

[8] Michael D. Adams *JasPer Software Reference Manual (Version 1.900.0).* ISO/IEC 2006 (see http://www.ece.uvic.ca/ frodo/jasper)

[9] C implementation of JPEG 2000 *openjpeg.org*

[10] E. C. Larson and D. M. Chandler *Most apparent distortion: full-reference image quality assessment and the role of strategy* Journal of Electronic Imaging, 19 (1), March 2010

[11] ITU-R *ITU-R BT.500-13 Methodology for the subjective assessment of the quality of television pictures.* 2012

[12] ITU-R *ITU-R BT.710-4 Subjective Assessment Methods For Image Quality In High-Definition Television.* 1998

[13] ITU-R *ITU-R BT.814-2 Specifications and alignment procedures for setting of brightness and contrast of displays.* 2007

[14] Sveriges Television AB *The SVT High Definition Multi Format Test Set.* Version 1.0 February 2006

[15] ffmpeg multimedia codec *ffmpeg.org* Version N-33308-g6638207 (built on the 5th of October 2011)

[16] Image magick image manipulation software *imagemagick.org* Version 6.7.5

[17] Videolan media player *videolan.org*

Appendix A

Subjective test

A.1 Results from subjective test

Bitrate [Mbits/s]	25	20	15	10	6	4	2
ParkJoy	0.98893	0.96882	0.92653	0.89572	0.83619	0.77480	0.69870
OldTownCross	0.99328	0.98298	0.97593	0.96152	0.94034	0.90986	0.86582
IntoTree	0.99189	0.97980	0.94957	0.91503	0.87656	0.83161	0.79290
DucksTakeOff	0.99032	0.98051	0.96776	0.93048	0.90036	0.82988	0.76844
CrowdRun	0.98734	0.97473	0.95258	0.91591	0.86639	0.80936	0.74651

Table A.1: SSIM values for the frame with the drop in bitrate, per bitrate. Calculated with 30 Mbits/s as the reference.

Bitrate [Mbits/s]	25	20	15	10	6	4	2
ParkJoy	0.98885	0.96402	0.92601	0.89649	0.83036	0.77428	0.70125
OldTownCross	0.99315	0.98278	0.97527	0.96105	0.93908	0.90863	0.86203
IntoTree	0.98808	0.97022	0.94073	0.90352	0.85535	0.80853	0.76674
DucksTakeOff	0.99222	0.98331	0.97144	0.94498	0.91187	0.85183	0.78543
CrowdRun	0.98714	0.97652	0.95410	0.91866	0.87024	0.81611	0.75209

Table A.2: Mean SSIM values over the entire source clips per bitrate. Calculated with 30 Mbits/s as the reference.

Bitrate [Mbits/s]	25	20	15	10	6	4	2
ParkJoy	34.130	30.714	28.308	25.468	23.273	21.859	20.683
OldTownCross	41.669	38.718	36.204	32.765	30.120	28.294	26.518
IntoTree	43.381	38.427	36.354	33.738	31.518	29.867	28.696
DucksTakeOff	35.808	31.985	29.409	26.841	24.457	23.229	22.051
CrowdRun	34.865	31.731	29.206	26.218	23.929	22.477	21.121

Table A.3: PSNR (Luma) values for the frame with the drop in bitrate, per bitrate. Calculated with 30 Mbits/s as the reference.

Bitrate [Mbits/s]	25	20	15	10	6	4	2
ParkJoy	34.192	30.853	28.060	25.380	23.131	21.815	20.645
OldTownCross	41.694	38.720	36.177	32.823	30.107	28.285	26.468
IntoTree	41.600	38.245	35.752	33.338	31.154	29.576	28.425
DucksTakeOff	38.024	33.847	30.722	28.064	25.323	23.719	22.137
CrowdRun	35.099	31.833	29.464	26.396	24.099	22.601	21.217

Table A.4: Mean PSNR (Luma) values over the entire source clips per bitrate. Calculated with 30 Mbits/s as the reference.

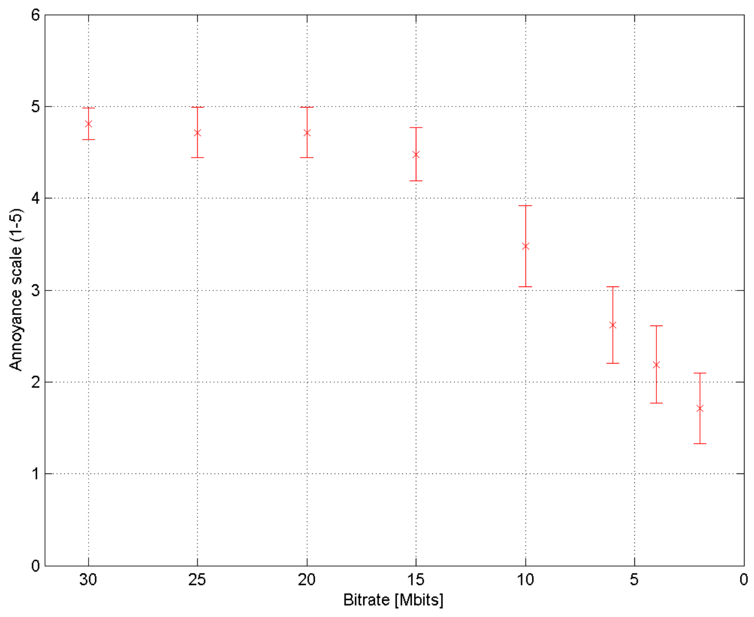


Figure A.1: Mean grades per bitrate for the ParkJoy clip. The line segments marks the 95 % confidence intervals.

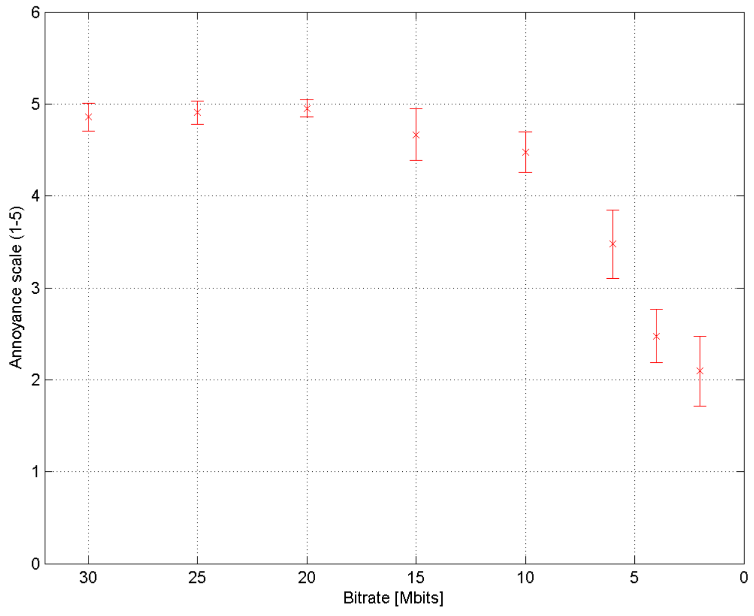


Figure A.2: Mean grades per bitrate for the DucksTakeOff clip. The line segments marks the 95 % confidence intervals.

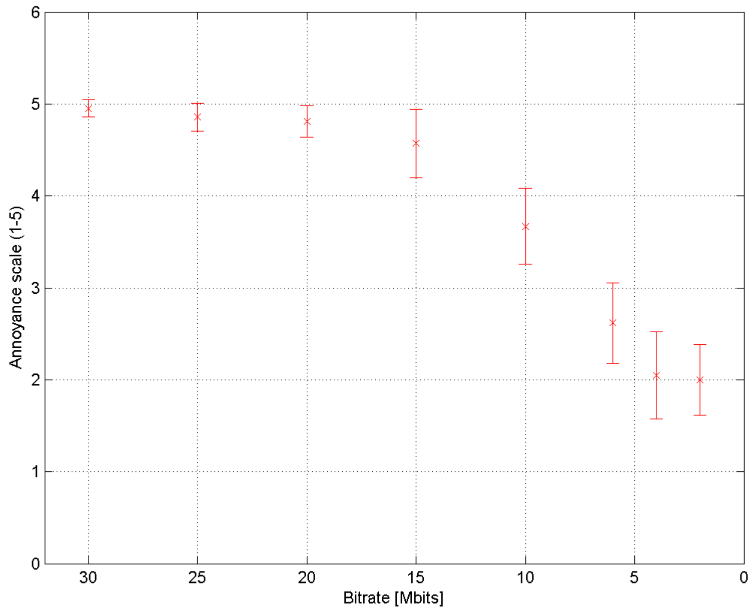


Figure A.3: Mean grades per bitrate for the InToTree clip. The line segments marks the 95 % confidence intervals.

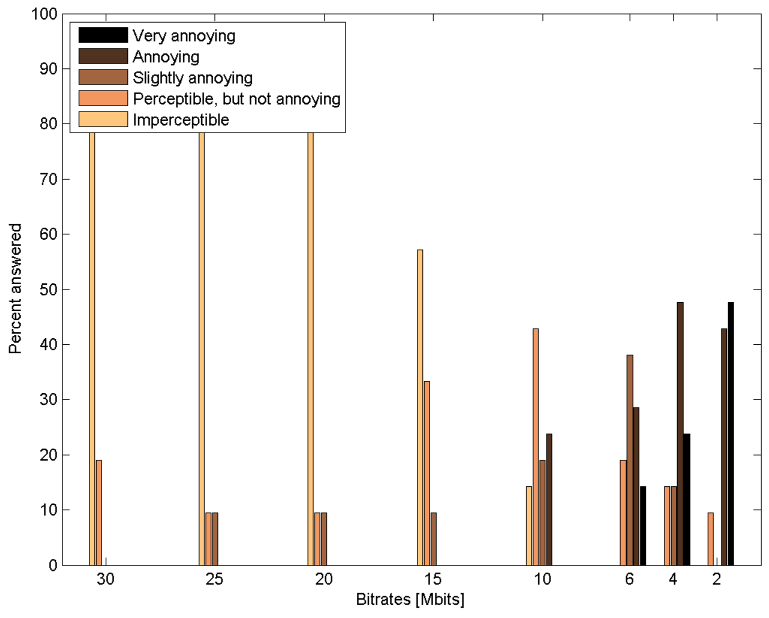


Figure A.4: Distribution of votes per bitrate the ParkJoy sequence.

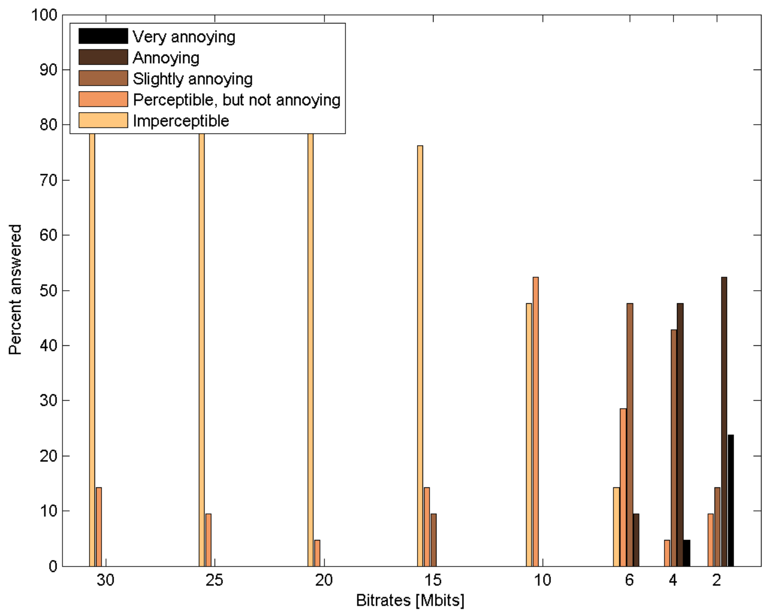


Figure A.5: Distribution of votes per bitrate the DucksTakeOff sequence.

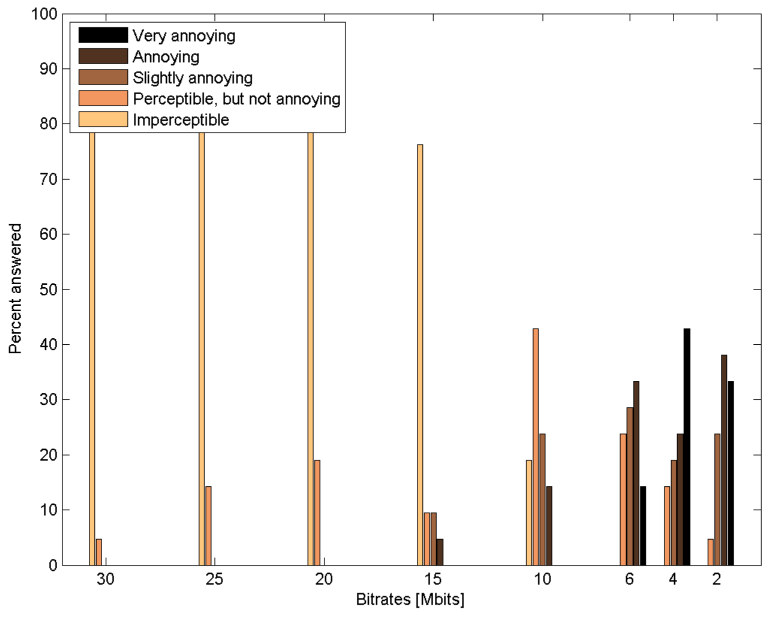


Figure A.6: Distribution of votes per bitrate the InToTree sequence.

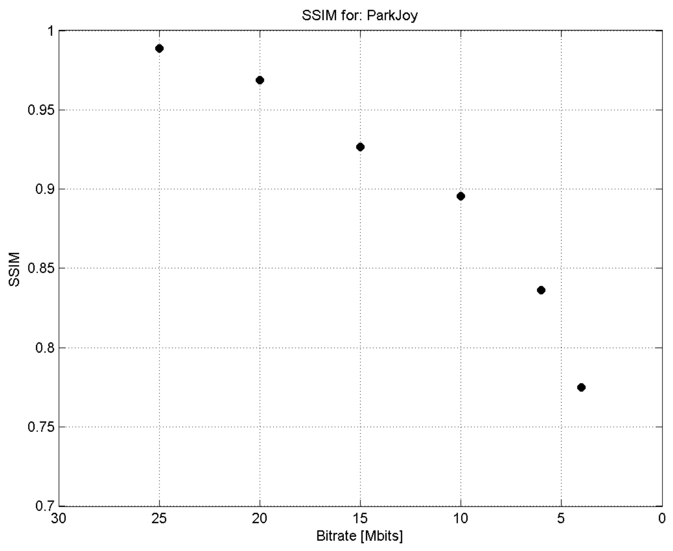


Figure A.7: SSIM index with respect to the reference bitrate for the ParkJoy clip.

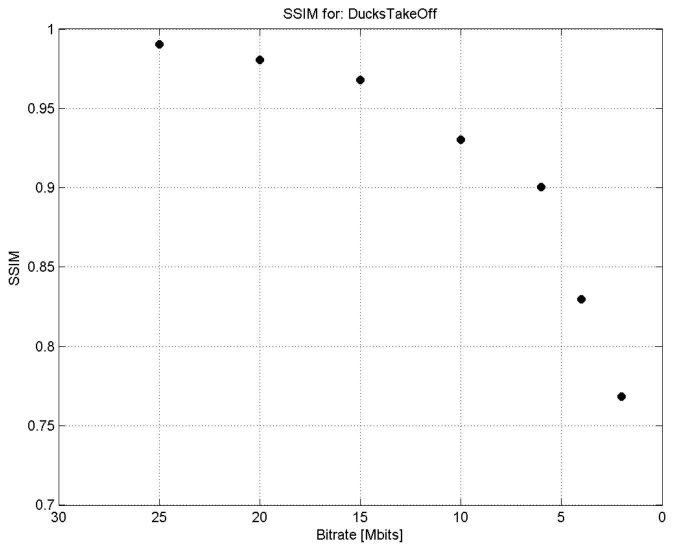


Figure A.8: SSIM index with respect to the reference bitrate for the DucksTakeOff clip.

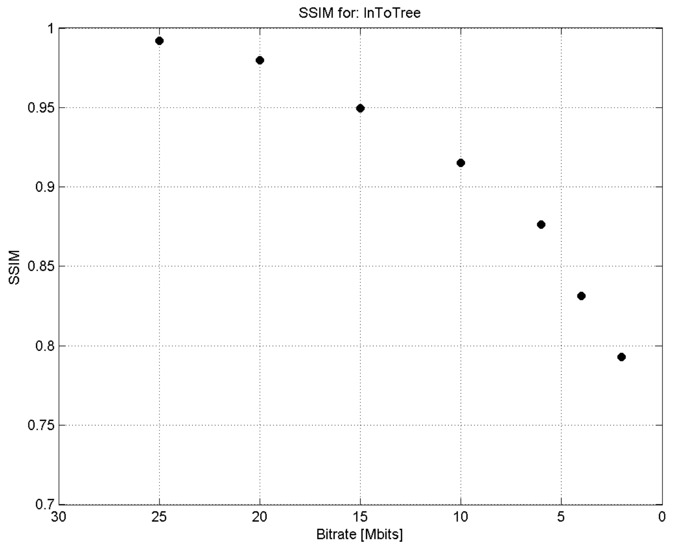


Figure A.9: SSIM index with respect to the reference bitrate for the InToTree clip.

A.2 Research protocol

Research protocol Author: Magnus Jeffs Tovslid

Synopsis

JPEG 2000 scalability can be used at nodes in a video transmission line or network to adapt the video bitrate for further transmission. This concept has been studied by [1], but subjective results are sparse. In order to make a sensible rate control algorithm at a transmission node, one needs information on how it will impact the subjective quality. This study aims to find out more about the subjective quality in such a system.

A subjective test will be run, where a reference will be compared to seven different tests. Each test will consist of a 10 second video clip where the bitrate suddenly drops to a smaller amount, and gradually increases again. The participants will be asked to rate the test on an annoyance scale (DSIS from ITU-R BT.500 [2]). In each of the seven test clips, the bitrate will drop by different amounts. The highest bitrates in the tests will be 40 Mbits/s, which will also be the bitrate of the reference clips. The lowest bitrate will be 15 Mbits/s. In between these bitrates, there will be 20 available quality layers, equally spaced in terms of bitrate. All the bitrates used in the test will correspond to such a quality layer. All together, five different source sequences will be used, which yields a grand total of 40 test clips over 20 minutes.

The outcome of this test will give insight into how a sensible rate control algorithm using JPEG 2000 scalability should be made. As an example, at what point should one leave the bitrate alone, and solve the problem with FEC, buffering, retransmission etc? The results of the subjective test will be analyzed using the standardized statistical procedure described in ITU-R BT.500 [2]. The annoyance rating in the different drops in bitrate will be compared to objective quality predictions at the same bitrates, to see if they follow a similar (or dissimilar) curve.

Background

JPEG 2000 quality scalability makes it possible to encode a bitstream once, and later extract several different rates/qualities from that stream. This makes it possible to have an encoder at a base station, and a quality layer-extractor at different nodes down the transmission line or network, which can determine the best number of quality layers to use for further transmission. Already, a similar scenario has been tested on a network simulator in [1]. This study utilized 8 different quality layers, and varied the number of layers for further transmission by using the TCP-friendly rate control (TFRC) and Video Transport Protocol

(VTP). However, there was no detailed discussion of why 8 layers has been chosen, nor was there any discussion on the subjective perception of varying the bitrate blindly according to the TCP-friendly rate.

The current study investigates how JPEG2000 quality scalability can be used to vary the transmission bitrate, and what kind of subjective results can be expected when using this approach. In particular, in order to design a good rate control mechanism at transmission nodes, it is necessary to investigate the subjective quality perceived when varying the bitrate using the quality scalability technique. In the current study, a technique for finding an optimal number of layers has already been found, and will serve as a basis for the subjective tests.

The subjective tests will mimic situations where the bitrate suddenly drops, and gradually goes back up again. The target bitrates will be approximated by extracting the appropriate number of quality layers from given video clips. The results will give an indication on how sensitive the human visual system is to changes in bitrate in JPEG 2000 intraframe video coding when using scalability as the rate control. The results from this test will be very useful when using scalability for e.g. channel capacity adaptation, as it will give an indication on what kind of drops in bitrate are acceptable to the viewer. Furthermore, this will give insight into what kind of situations should be handled by other techniques, such as FEC and buffering.

Hypothesis

- It is expected that the level of annoyance will increase with the size of drops in bitrate, and that there is a point where the level of annoyance goes from "perceptible, but not annoying" to "slightly annoying".

Methodology and design

Resources required
- The Café Media room at NTNU is required for approximately 5 days for testing
- Up to 25 people will participate in the test, and as a reward for participation they will each
receive a cinema ticket. These tickets are the only needed funding for the project.

Construction of the test

In the test, the degree of annoyance experienced in drops in bitrate will be measured. The test is a subjective one only. It will employ the ITU-R BT.500 standard for subjective testing. The double stimulus impairment scale (DSIS) method will be used, since it is the degree of annoyance that is the interesting factor in this study. The test itself will last 20 minutes, along with up to 10 minutes of training and introduction. Since each test in the DSIS method last roughly 30 seconds, this leaves enough time for 40 tests.

Details

- The test will consist of a 20 minute long video clip, generated by ffmpeg by concatenating encoded JPEG 2000 frames. The clip will be the same for all participants, since the file size makes it impractical to change for each participant.
- The participants in the test will receive written instructions for how the test shall be done.
- The source material for the test will be from the SVT fairytale set. This set consists of five different 10 second clips of very high quality (uncompressed).
- A total of 40 tests divided by 5 test clips gives a new total of 8 tests. Among these, one will just be the reference clip itself, so that leaves 7 tests. This means that 7 different amounts of bitratedrops will be tested.
- Each frame to be sent through the layer-extractor has been prepared such that it contains 20 equally spaced layers in the range 15 to 40 Mbits/s.
- In all the clrops in bitrate, the bitrate will start at 40 Mbits/s, which is considered of good quality for normal viewing (but not flawless). In the middle of the test clip, the bitrate will drop a certain amount, and linearly increase back up to 40 Mbits/s over a time period of 2 seconds. By linearly, it is meant that the bitrate fed into the quality layer extractor will increase linearly, but the actual bitrate will increase in steps defined by the layers. The layers, however, are equally spaced in terms of bitrate.
- The bitrate-drops to be used should ideally represent the entire range of impairments inside the chosen bitrate range. However, since there is only time for 7 different drops in the test, the result will only be an approximation. The following amounts of drops in bitrate have been chosen (in Mbit/s) (subject to change):
3.5 7 11.5 15 18.5 22 25

Results and analysis

In the ITU-R BT.500 standard, a statistical method for analyzing the results of a DSIS test is given. In this method, mean values are calculated and checked for statistical significance according to a 95 confidence interval. The method also includes a check for normally distributed results. The mean values calculated will then be used in a comparison with objective quality metric values (SSIM and PSNR). The main question to be answered here is: Will the values fall off along a similar curve, or are there differences? If they follow a similar curve, then it might indicate that varying the bitrate directly according to the rate will be a good idea. The main problem with this type of comparison is that the annoyance scale and the quality scale produced by the metrics are not the same. Therefore only the shape of the curves can be compared. The results will be used in my master thesis.

References

[1] "Adaptive Rate Control of Motion-JPEG2000 video over IP networks" Rong-Yu Qiao and Michael H. Lee

[2] "Methodology for the subjective assessment of the quality of television pictures" ITU-R BT.500-13 (01/2012)

A.3 Pictures from the testing room

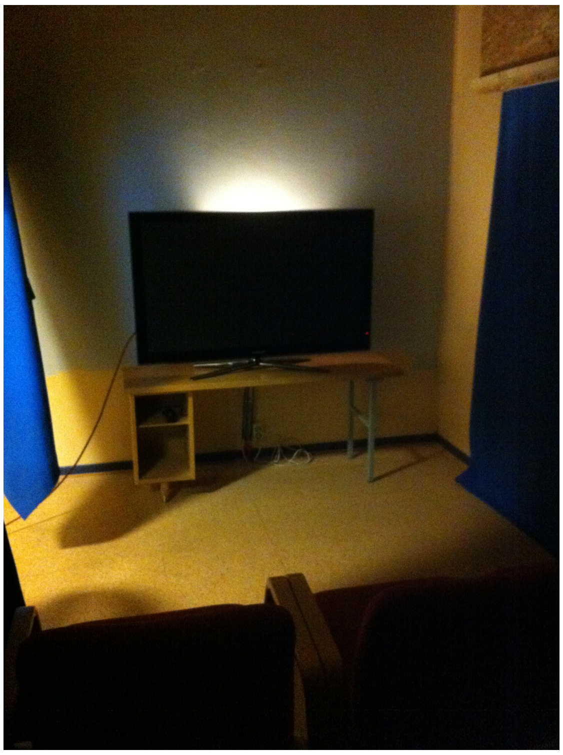


Figure A.10: Picture taken from behind the chairs.

Figure A.11: Picture taken from outside the testing area.

Figure A.12: Picture of the three chairs in the testing area.

A.4 Instructions given to the participants

Description of subjective test procedure

The test you are about to participate in will consist of all together 45 test video clips. In some test clips, the video quality will rapidly fall by a certain amount, and then gradually increase back up again. This will take place somewhere in the middle of each clip. A test clip will last for 10 seconds. Your job is to rate the quality degradation in the test clips on the following scale:

Grading scale:

1. Imperceptible (Norsk: Ikke merkbart)
2. Perceptible, but not annoying (Norsk: merkbart, men ikke irriterende)
3. Slighly annoying (Norsk: litt irriterende)
4. Annoying (Norsk: irriterende)
5. Very annoying (Norsk: veldig irriterende)

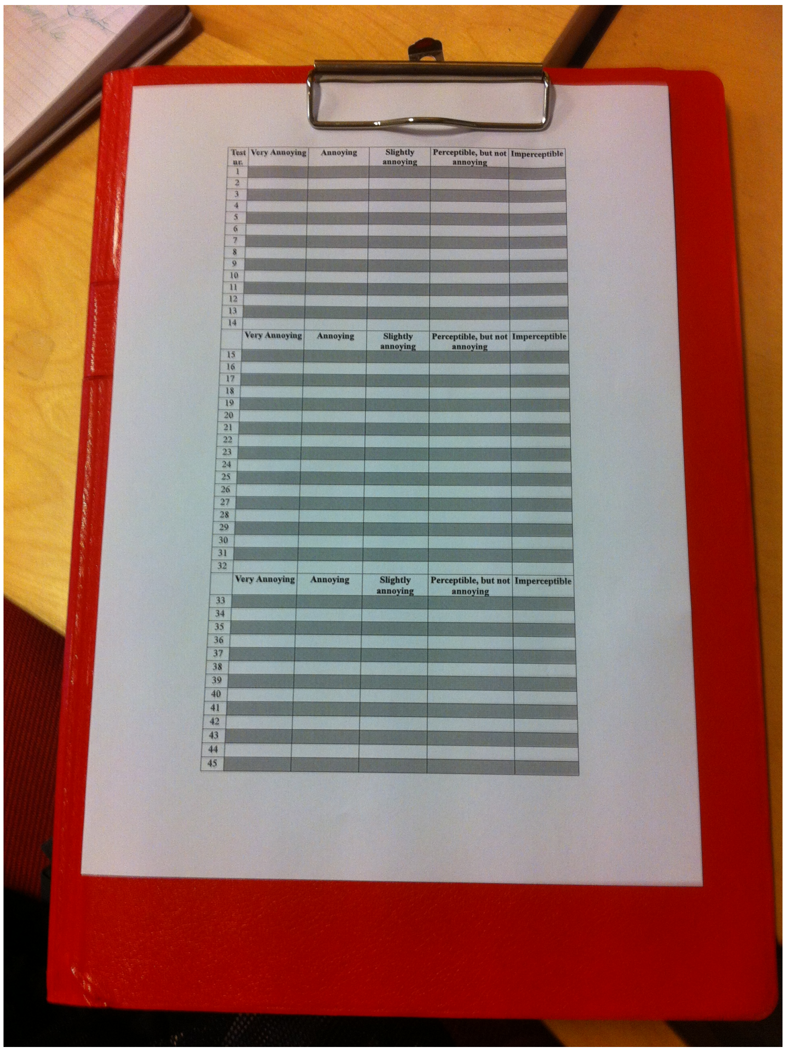
It is important that you are totally honest in your ratings. Answer based on your overall impression of the test video clip, and not on what you think someone else might say. Before each test clip, you will be shown the same clip without any quality degradation. In between this reference clip and the test clip, there will be 3 seconds of grey screen. After both the reference and the test clip have been shown, 7 seconds of grey screen will be shown, with a number indicating which test clip you just watched. During these 7 seconds, you should write down your vote on the voting paper.

Test timing:

10s reference clip (the same as the test clip, but without any quality degradation)
3s grey screen
10s test clip
7s grey screen with a number indicating which test clip was just shown - give your vote here

The test will take a little over 22 minutes to complete.
Before you start the test:
- Fill in the information on the next page.
Before you leave:
- Grab a cinema ticket
- Sign your name

A.5 Voting paper



Test nr.	Very Annoying	Annoying	Slightly annoying	Perceptible, but not annoying	Imperceptible
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
	Very Annoying	Annoying	Slightly annoying	Perceptible, but not annoying	Imperceptible
15					
16					
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					
	Very Annoying	Annoying	Slightly annoying	Perceptible, but not annoying	Imperceptible
33					
34					
35					
36					
37					
38					
39					
40					
41					
42					
43					
44					
45					

Figure A.13: Picture of the voting paper used by the participants in the subjective test.