NTNU
Norwegian University of
Science and Technology

Wireless Video Capsule Endoscopy

Adaptive Coding for Parameters Update and
Low Complexity Region-of-Interest Encoding

Endre Våland Bø

Master of Science in Electronics
Submission date: June 2016
Supervisor: Ilangko Balasingham, IET
Co-supervisor: Øyvind Janbu, Oslo University Hospital

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Endre Våland Bø

Wireless Video Capsule Endoscopy

Adaptive Coding for Parameters Update and Low Complexity Region-of-Interest Encoding

Master of Science in Electronics

Trondheim, June 2016

Supervisor: Ilangko Balasingham, IET

Co-supervisor: Øyvind Janbu, Oslo University Hospital

Norwegian University of Science and Technology

Faculty of Information Technology, Mathematics and Electrical Engineering

Department of Electronics and Telecommunications

NTNU

Norwegian University of Science and Technology

Abstract

Wireless capsule endoscopy (WCE) is a non-invasive diagnostic method to investigate diseases in the human gastrointestinal (GI) tract. Specifically, the WCE allows for physicians to visually inspect the GI tract of the patient who swallows the capsule. Thereafter, it transmits video wireless from the inside of the human body to the outside, for approximately eight hours. Because of the size and the limited battery of the capsule, the image quality can be quite poor. By reducing the average bitrate per pixel, the energy consumption will be reduced, which will allow higher quality images, higher framerate or resolution.

When applying the YEF colour transformation instead of the YUV, as well as adaptive prediction, the rate can be reduced by circa 0.12-0.35bpp. Additionally, this resulted in an increase in peak signal-to-noise ratio (PSNR) and Structural Similarity Index (SSIM), depending on the input simulation video. On the other hand, these results were very dependent on the source video and might by different in a real implementation. This should be investigated further with simulation videos more similar to the ones captured by the camera sensor on the capsule.

Another possible solution to improve the overall image quality is to encode some regions of the image in higher quality than the rest of the frame, known as region-of-interest (ROI) coding. A very low complexity, energy efficient, ROI coding scheme was proposed for the encoder (the capsule). The method relies on adaptive sampling rate within a frame, to create a region which isn’t down-sampled or filtered. Simulations showed that it works as expected in the encoder, but it does have some problems at the decoder. This should be investigated further, as well as to correctly examine the increase in energy consumption in the proposed method.

Sammendrag

Trådløs kapselendoskopi (WCE) er en diagnostiseringsmetode innen medisin for å undersøke menneskets mage- og tarmsystem, for å lete etter skader eller sykdommer. Pasienter svelger en kapsel på størrelse med en vanlig pille, som tar bilder mens den beveger seg gjennom systemet. Bildene blir trådløst overført til en mottaker som sitter på et belte på utsiden av kroppen. Ettersom pillen er liten og skal vare i omtrentlig åtte timer, er fysisk størrelse og batterikapasitet begrenset. Disse begrensningene medfører at mottatt video er av utilfredsstillende kvalitet. Ved å redusere gjennomsnittlig datarate per piksel, vil energiforbruket reduseres, som tillater høyere kvalitet på bildene, høyere bildesekvens eller oppløsning.

Ved å benytte YEF fargetransformasjon istedenfor YUV, i tillegg til adaptiv prediksjon, ble gjennomsnittlig datarate redusert med mellom 0.12-0.35bpp, avhengig av simuleringsvideo. Samtidig økte signal-til-støy forholdet (PSNR) samt strukturell likhetsindeks (SSIM). Resultatene var tydelig avhengig av kilden, siden bildekompresjon av komprimert bilde økte objektiv evaluering, så det er forventet at resultatene ikke vil være identiske i en faktisk implementasjon. Dette bør undersøkes videre ved bruk av simuleringsvideoer som er identiske med det som produseres av bildesensoren på kapselen.

En annen løsning for å øke den overordnet bildekvaliteten hos mottakeren kan være å kode enkelte deler av bilderammen med høyere kvalitet enn andre, også kjent som interesseområdekoding (ROI). I denne sammenheng ble det foreslått en lavkompleksitet- og energieffektiv ROI implementering for koderen (kapselen). Metoden innebærer å forandre punktprøvingsfrekvensen innad i bildet, som skaper en region som ikke reduserer punktprøvingsfrekvensen eller lavpassfiltrerer verdiene. Simuleringene viste at dette virker som forventet på enkoder siden, men det er noen problemer på dekoder siden, samt i ROI evalueringene. Dette må videreutvikles slik at foreslått metode kan bli korrekt evaluert, samt undersøke hvor mye energiforbruket øker med denne kodingen sett sammen med økt kvalitet.

Preface

This thesis is the finalization of the Master of Science (M.Sc.) degree at Norwegian University of Science and Technology (NTNU). The work is based on a specialisation project conducted in the fall 2015, and the thesis work was completed the spring 2016.

Working with the wireless capsule endoscopy has been interesting and exciting, but also frustrating and challenging. Extensive research has been conducted a lot of research all over the world within wireless capsule endoscopy technology. It has been a major motivating factor to be a (small) part in this vast network of researchers who continuously try to improve the diagnostic abilities for the physicians.

I need to give a big thank you to my supervisor Ilangko Balasingham. He has given me many valuable advises, and a lot of good directions/discussions to get me on the right working path of the thesis. The motivation, insight and guidance he has given me, has been very important to achieve the resulting product.

I must also thank Øyvind Janbu, for giving good recommendations and providing opinions in what he suggested for the thesis at an early stage, which was very motivating and helpful.

Lastly, I would like to thank all of my friends and family members for great support throughout the project. Especially my fiancée, Helene Doublet, deserves a big appreciation for her support, motivation and her encouragement through the thesis.

Endre Våland Bø

Trondheim, 21.06.2016

Table of Contents

List of Figures

List of Tables

Abbreviations

ADC	Analogue-to-Digital Converter
AVI	Audio Video Interleave
BA	Backward Adaptive
BPP	Bit-Per-Pixel
CMOS	Complementary Metal–Oxide–Semiconductor
CPSNR	Colour Peak Signal-to-Noise Ratio
CR	Compression Ratio
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DWT	Discrete Wavelet Transform
DZ	Dead-Zone
EM	Electromagnetic
ERP	Effective Radiated Power
FA	Forward Adaptive
FPS	Frames Per Second
GI	Gastrointestinal
GOP	Group Of Pictures
HP	High-Pass
IEEE	Institute of Electrical and Electronics Engineers
JPEG	Joint Photographic Experts Group
LAR	Locally Adaptive Resolution
LED	Light Emitting Diode
LP	Low-Pass
MOS	Mean Opinion Score
MPEG	Movie Picture Expert Group
MR	Multirate
NB	Narrow Band
PL	Path Loss
PSNR	Peak Signal-to-Noise Ratio
RLE	Run-Length Encoding
ROI	Region-Of-Interest
SBC	Subband Coding
SNR	Signal-to-Noise Ratio
SQNR	Signal-to-Quantization-Noise Ratio
SR	Stack-Run
SSIM	Structural Similarity Index
URURQ	Uniform Reconstruction with Unity Ratio Quantizer
UTQ	Uniform Threshold Quantizer
UWB	Ultra Wide-Band
WCE	Wireless Capsule Endoscopy

1 Introduction

There has been conducted a lot of research within minimization of medical equipment, over the past years. One of these fields is the wireless capsule endoscopy (WCE). WCE allows for medical examinations of the gastrointestinal (GI) tract to diagnose diseases like cancer, tumour, Crohn's disease, and bleedings to mention a few [1, 2]. The capsule is swallowed by the patients, and is then moved through the GI tract by the natural contractions in the tract (peristalsis), the exactly same way as eaten food is [3].

Various versions of the WCE exist today, but what is common among these are that they usually consist of a CMOS camera sensor with LED, microchip, battery and a radio frequency (RF) transmitter [4]. The patient wears a sensor belt with an array of receiver antennas and a small computer to decode and store the captured images [5]. However, since the capsule is so small, the battery capacity is very limited, resulting in quite poor image quality, making diagnosing harder for the physicians.

Because of this limitation in power availability, increasing the resolution, frame rate, or image quality (minimizing distortion) is difficult. A possible solution is to reduce the required energy to encode the frames. This can be achieved by analysing the algorithm to make it use less bits per pixel, in order to increase the received image quality.

Another possible solution to get a higher image quality could be to encode some regions of the image at a higher quality than the rest, also known as region-of-interest (ROI). ROI encoding is a known image encoding method, but require more from the encoder (in terms of energy consumption), which is unwanted in this case. Therefore, it would be advantageous if a very low complexity ROI encoding scheme could be implemented in the WCE case.

Figure 1.1 PillCam SB3 from Given Imaging. Most widely used model from Given Imaging [1].

1.1 Motivation

Normal endoscopic procedures are quite commonly used when diagnosing (or treating) diseases in the GI tract, but this can be very uncomfortable for the patients [6, 7]. On rare occasions the endoscopic equipment might even scratch the walls in the tract, causing bleedings and potential infections. The wireless capsule endoscopy, on the other hand, isn't felt by the patients at all, it is safe, thus making it preferable to the former for the patients.

Physicians rely on the image quality to be able to make the diagnosis, and since normal endoscopy (not WCE) have a higher image quality, this is preferred as long as it can reach the area[1]. By improving the image quality, both the patients and the physicians can be satisfied in both comfort and diagnosis. This isn't easy, since improving the quality generally involves increasing the complexity which requires more energy [8]. Some work has been done in attempt to increase the available power by wireless power transfer using ultra-wideband (UWB) or magnetic resonance technology [9-12], but this is still at quite early research stage. So at the time, improving the image compression scheme would be a good way of improving the image quality.

This thesis is mainly based on two different works. Firstly, a project conducted in the fall of 2015 which briefly investigated the possibility of adapting parameters, and secondly the original algorithm proposal [7, 13]. These are explained more in Chapter 3.

1.2 Objective and Limitations

The aims of this thesis are to explore the possibilities of a feedback loop, and to adapt the WCE during the active period through the GI tract. In a feedback loop the coefficients can be adapted according to the position of the WCE, allowing for a higher quality image. To achieve this, a study of the current algorithm as well as other image encoding techniques will be conducted to find potential improvement areas.

A feedback loop also enables the possibility of ROI encoding without increasing the required computational operations. Analysing the received frames at the decoder can give potential ROI's, where the positions are transmitted back. This ROI area can then be encoded using higher quality than the rest.

Note that the WCE can have many different sensors attached to it. These can for instance be measuring the pH parameter, pressure information, temperature or video capturing [14]. This

[1] The small intestine isn't reachable with normal endoscopy, so for this area WCE is preferred, see Section 2.1.

thesis is limited to concern the video or image capturing capsule, and focus to improve the image compression algorithm. Improvement of the algorithm can include to reduce the amount of data per transmission or reduce the complexity of the encoding algorithm, in order to save energy. Reduction of transmitted data, will allow for additional sensors and/or improvement of quality, since the available power is constant. A key aspect is to avoid increasing the complexity of the algorithm, but since there is always a trade-off between complexity and received quality this might be a challenge [15].

The problem will be limited to a simple implementation, and verification with simulations. The results are therefore expected to differ from a real implementation in a device, but will be compared with theory and existing research to be able to draw a conclusion based on the findings.

1.3 Structure of the Thesis

The thesis is theoretical, so a comprehensive background will be presented in Chapter 2. This chapter will include theory on common image encoding techniques before the original scheme for the WCE is presented in Chapter 3.

Based on the theoretical background presented in Chapter 2 & 3, the proposed scheme will be presented in Chapter 4. This chapter also elaborate the implementation and the chosen evaluation methods.

In Chapter 5, the simulation results from the proposed algorithm improvements and the ROI coding, is presented. The results will then be discussed in Chapter 6, where a comparison with existing algorithms and possible error sources, will be studied. In this chapter possible hardware implementation will briefly be examined before a summary or conclusion together with future work, will be presented in Chapter 7.

Lastly a bibliography of the analysed literature is given in Chapter 8, before the appendixes. In Appendix A, the different programs used throughout the thesis work is listed in A.1, together with specifications in the simulation videos is given in A.2, before the MATLAB code is given in Appendix B.

.

2 Theory

In this chapter, the necessary theoretical background will be presented, to establish a fundament for how the algorithm works, and what can be done with the limitations. Chapter 2 is a general presentation of compression techniques and a short medical background. After this, a more specific theory section of the wireless capsule endoscopy is presented in Chapter 3. This will be the basis for the choices done in order to improve the image quality with the application specifics limitations.

2.1 Medical Background

Firstly, a brief medical background will be presented in this section, to understand how the human anatomy works (Subsection 2.1.1-2.1.2) and how a general biomedical image processing system can help diagnose patients (Subsection 2.1.3).

2.1.1 Human Anatomy – The Gastrointestinal Tract

The human digestive system consists of the gastrointestinal (GI) tract, liver, pancreas and gallbladder [3]. The GI tract consists of hollow organs stretching from the mouth to the anus. This includes the mouth, oesophagus, stomach, small- and large intestine, the rectum and the anus, as shown in Figure 2.1. The systems' main task is to breaks down the food to different nutrients in which the body needs for energy, growth and cell repairs. Food enters the mouth, and passes through this hollow GI tract, before going out through the anus. The food is broken down by two main parts of the digestive system; the mechanical and the chemical.

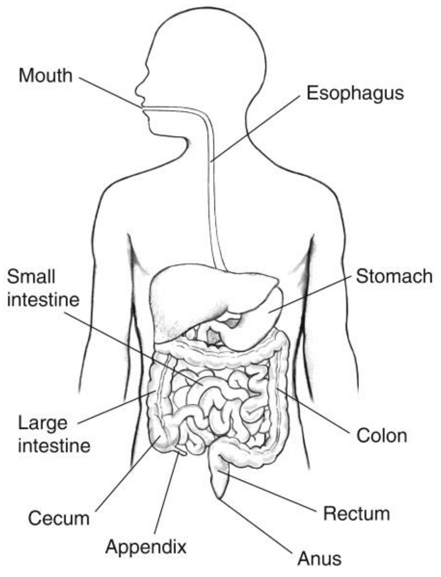

Figure 2.1 The GI tract [16].

The mechanical part is also known as chewing, when the food enters the mouth at the beginning of the GI tract. The chemical part is happening along the whole way through the GI tract. Different digestive juices are mixed with the food to break down food particles such as starches (saliva), proteins (stomach acid), carbohydrates (small intestine digestive juice) and fats (pancreatic juice & Bile acids) [3, 17]. Digestive juice contains enzymes which speeds up the chemical reactions, and together with bacteria (also called gut flora or microbiome) in the GI tract, the food is broken down. This food becomes small molecules in which the body absorbs through the wall in the small intestine, and into the bloodstream. The bloodstream carries these nutrients to the whole body, where they are needed. Waste, also called stool, from the process is passed along the GI tract and out the anus.

All of the hollow organs in the GI tract contains a layer of muscles, which allows for the walls to move (muscle contraction) [3, 18]. These muscle contractions make muscular waves, also called *peristalsis*, that travels the whole length of the GI tract. Peristalsis propels the food and liquid through the organ, while activating the production of different digestive juices along the way. Along with the peristalsis there are two other main processes for movement and mixing of the food; swallowing and segmentation. Swallowing uses smooth and skeletal muscles in the mouth, tongue and pharynx[2] to push the food through the pharynx and into the oesophagus. Segmentation occurs in the small intestine and is short contractions which squeezes the food, for improving the absorption of the small molecules into the blood stream.

The whole digestive process is controlled by two regulators; hormone and nerve regulators [3]. The hormone regulator is produced from cells in the stomach and small intestine. These hormones stimulate production of digestive juices and controls the appetite. Nerve regulators controls the action of the GI tract, and there are two kinds of nerve regulators; extrinsic (outside) and intrinsic (inside). Extrinsic nerves release chemicals to make the muscles in the GI tract to contract or relax. Intrinsic nerve regulators release different substances to speed or delay the movement of food in the GI tract, and are triggered when the food stretches the walls.

The GI tract is a complex system with many different organs working together to digest food and produce energy for the whole body. As with every complex system, it can also be vulnerable to different diseases and injuries. Some of these will briefly be introduced in the next section, before some of the diagnostics methods will be presented.

[2] Pharynx is part of the throat, which can close off in various ways for speaking, breathing, swallowing etc. [19]

2.1.2 Diseases

A digestive disease is any health problems that can occur in the GI tract, and is very common either as a mild or as a serious disease [20]. Since the GI tract consists of four distinct parts (oesophagus, stomach, small- and large intestine) with different functions to perform the digestion, each part has a unique type of motility (contractions) and sensation [21]. These parts are separated by sphincter muscles[3], and will in turn can give many distinct diseases and symptoms. Functional and motility disorders are the most common GI disorders among the general population. There is some uncertainty about how many is affected, but a study showed that 42% of the population was affected over a 12 years' period. However, another study showed that 75% of the population which experienced some of these symptoms, didn't consult for medical care [22].

In general, the term "functional" disorder relates to disorders where some or more of the body's normal actives are impaired [22]. These actives can be in terms of intestine movement, nerve sensitivity or the way the brain controls some of these functions. Common among these disorders, is that they don't have any structural abnormalities which can be seen by common diagnostic methods, including endoscopy, x-ray and blood tests [23]. Instead they are diagnosed based on the characteristic symptoms and sometimes limited tests. Some examples of these disorders can be functional diarrhoea, functional vomiting, irritable bowel syndrome (IBS) or functional abdominal pain, to mention a few.

"Motility" is defined as the movement of the digestive system and the content in it [24]. Disorders regarding motility occur when any nerves or muscles in any part of the GI tract don't function with normal strength or coordination. Unlike the functional disorders, the motility ones can be diagnosed using methods as for instance endoscopy or oesophagram. For instance, some disorders include gastroesophageal reflux disease (GERD), constipation or small bowel bacterial overgrowth [25].

Many possible GI disorders don't fit into functional or motility disorders, but are still common [20, 26]. These other diseases can have symptoms that are similar to those belonging to the functional or motility disorders. However, these can be uniquely identified since they have some unique feature, depending on the kind of the disorder. Many different diagnostic methods can be used to identify these features, for instance x-ray, Magnetic resonance imaging (MRI),

[3] Sphincter muscles are special muscles that is normally tightly closed, and opens when food arrives, preventing food from going the wrong direction [21].

endoscopy among more. Typical disorders include cancer, lactose intolerance, bleedings, Crohn’s disease or other inflammatory diseases [16, 26].

2.1.3 Diagnostic Methods

Since there are a lot of different disorders that can occur in the GI tract, there are also many different methods for diagnosing diseases or injuries. Naturally, these methods depend on which symptoms and disorder a patient present. Together with the symptoms it is common for the physicians to run a laboratory test and or different imaging techniques to correctly diagnose a patient. Naturally, laboratory tests aren’t relevant in this case, so the focus will be on imaging techniques. Common imaging diagnosing techniques can include x-ray, colonoscopy, endoscopy, capsule endoscopy, magnetic resonance imaging (MRI), ultrasound or computed tomography scan (CT scan), to mention a few [27]. It is often common to use different imaging methods depending on where the symptoms occur. For instance, endoscopy and x-ray are common in the oesophagus and stomach, capsule endoscopy in the small intestine and colonoscopy in the rectum and large intestine. In general, all of these biomedical image techniques follows the same steps [28]. Some kind of sensors capture an image or signal from a biological system, before pre-processing and filtering to remove the unwanted noise. Next, extraction of the relevant features is done, to be able to describe the status before classification and diagnosis, as shown in Figure 2.2 below.

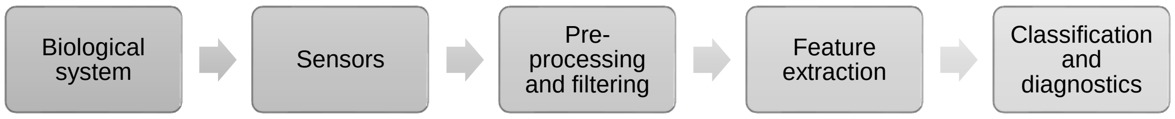

Figure 2.2 General biomedical image processing system [28].

Regular photographic images are captured by cameras (sensors) which captures the light intensity and/or colour of the objects [28]. Biomedical images on the other hand doesn’t necessarily capture the images in this way. For instance, MRI capture images by recording the magnetic properties of a tissue, and CT scan records many x-ray beams and the different interaction between themselves, and the tissue to form an image [29]. The result of these other kinds of recording, is that properties and functions of the tissues can be captured that normally wouldn’t be possible for humans to see.

Sometimes, the physicians need to be able to see inside a patient. In such cases, non-invasive methods, such as normal photographic images can be enough to be able to get a diagnosis of the patients. Diagnostic equipment for imaging includes endoscopy, capsule endoscopy, laparoscopy or colonoscopy [30, 31]. Most endoscopic procedures often involve a small camera[4] connected to the end of a flexible thin tube, which is inserted either through the mouth, anus or through abdominal cavity (laparoscopy). Capsule endoscopy is a little different from the other endoscopic procedures in the way that they are not connected to a tube. The sensor are instead mounted on a small computer, the size of a pill, which is swallowed, taking pictures along the way through the GI tract [27, 32]. This method is mainly used to detect bleedings, tumours, inflammatory diseases, polyps or cancer in the small intestine, where the other types of endoscopic techniques can't reach.

When diagnosing a patient using normal camera sensors, a useful tool can be the feature extraction part (Figure 2.2). In a typical endoscopic tool, the light source is xenon light, which have wavelength variating between 470-700nm [33]. Blood absorbs more light than surrounding tissue, because blood consists of about 45% red blood cells (and 55% plasma), which contains the oxygen carrying protein haemoglobin [34]. Therefore, variation in blood volume affects transmission and reflectance, which correspond to the thickness of the tissue analysed [35, 36]. This principle is called photo-plethysmography (PPG), and can be used to detect abnormalities in an area, compared to the surrounding tissue. If a picture from inside the GI tract contains a tumour or cancer, the absorption and reflection of the light will be different since the tissue is a lot thinner at this area. However, to be able to detect these possible abnormalities efficiently and reliable, high quality images is required. To achieve this a good image compression algorithm is needed, which will relate to the pre-processing and filtering part in the biomedical image processing system (Figure 2.2).

2.2 Image Compression

Every digital signal or image requires a lot of bits per second to transmit or many bits to represent in storage, which results in high costs [37]. For instance, an uncompressed image, with a size of 512x512 pixels, would require 512x512 pixel colour image x 24 bits/pixel = 6,3 Mbits per image. Thus image compression is applied to reduce the required bits (or costs). In general, data compression can be defined as minimizing the required bits needed to represent the source, while maintaining acceptable reconstruction of the source. Hence the storage and/or transmission costs are reduced. This can be done in many different ways, depending on the method used and the signal (application).

[4] There are other sensors which is used for this purpose as well and the procedure isn't limited to a camera. Other sensors can be pH meter, temperature meter, endoscopic ultrasound (EUS) or endoscopic retrograde cholangiopancreatography (ERCP) [32].

Many signals contain information which isn't needed, either because the information can be retrieved, or because it isn't relevant for the application. In redundancy removal, there are correlation between consecutive samples, which means some samples can be removed and reconstructed at the decoder, using prediction. With irrelevancy, some of the information in the signal can't be perceived by the user, meaning it can be removed without it being noticed. Redundancy is without loss of information (lossless), while irrelevancy is with loss of information (lossy).

2.2.1 Source Coding

When compressing a source, it is common to use the terms *lossless* and *lossy* compression algorithms [37]. Lossless means that perfect reconstruction is possible at the receiver, and the result is identical to the source. Lossy compression on the other hand, means that the source is not perfectly preserved and the reconstructed data is not identical to the original. It is important to remember that an algorithm can be lossless even if the result isn't identical. This is because the transmission of a signal or image will be affected by noise or interference which can alter the signal (more in Chapter 3.2). However, this is due to the *channel*, and not the compression algorithm. If the channel is ideal, the reconstructed signal or image would be identical to the source.

Naturally, lossless compression schemes have a higher bitrate than the lossy compression algorithms, since information is lost in the lossy case. However, the high bitrate in lossless means that there is no distortion[5] in the signal or image. When limiting the bitrate to a lower value than needed for the lossless compression scheme, distortion occurs to the signal. This is known as *rate distortion theory*, and considers the minimum bitrate which is required for a given image quality (or given fidelity) [39]. The relationship between bitrate and distortion follows a function, R(D), based on the lower bound for the transmission bitrate, shown in Figure 2.3 below.

[5] Distortion means to change the shape or appearance of a signal, in usually an unwanted way [38]. In practice this means that a different symbol is received than was sent.

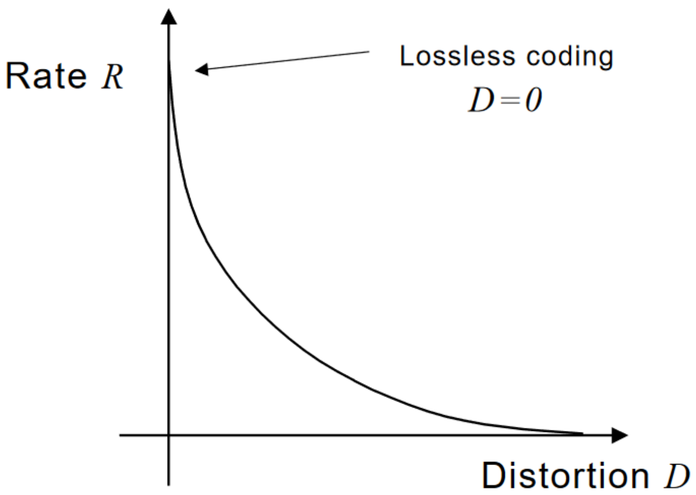

Figure 2.3 Rate-Distortion function [40]

This rate distortion function, or source coding theorem, is also known as one of the fundamental limitations (within source coding), and was derived by Claude E. Shannon in 1959 [8, 41]. It is important to remember that this only applies to lossy compression schemes. In practice, the lossy compression technique is the *quantization* part, as shown in Figure 2.4 below. The other two blocks are the lossless part of the general compression system. In other words, nearly every lossy compression scheme, contains a lossless part and a lossy part [39].

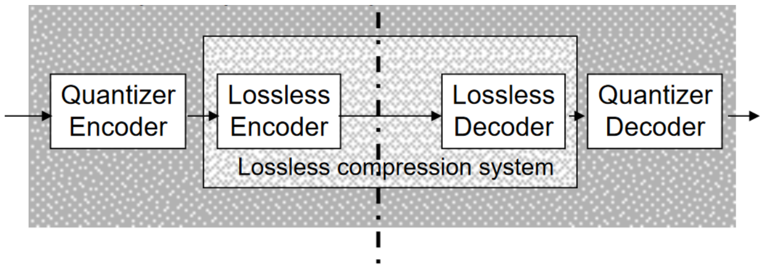

Figure 2.4 Lossy compression system [42]

The compression system is based on *Shannon's separation principle*, which states that for point-to-point communication systems, reliable transmission is possible if the source coding rate is below channel capacity [43]. This means that it is possible to look at the source coder and the channel coder separately, and still be able to see if it will work. However, this theory is under the assumptions of ergodic channel[6], and complexity and delay tends towards infinity. This is never the case in a practical system, especially in networks and in feedback systems,

[6] Ergodic channel has the property that given enough time, all samples in a given (small) space can be represented statistically by a large selection of samples.

resulting in errors in the received signal. The theory still gives an important principle; to optimize the whole communication system, one can optimize the source coder and channel coder separately. Received signal, for instance, can either be tolerated as sub-optimal and/or error correction methods can be applied (presented in Section 3.4).

2.2.2 Building Blocks for Image Compression

When optimizing the source coder is it, as mentioned, also common to separate into different steps (not to be confused with Shannon's separation principle), depending on the application. This can be different redundancy reductions (reversible) or irrelevancy reductions (not reversible). It is common to distinguish between three main steps, and two additional steps [44]. The first main step is the forward transformation. This is a redundancy reduction, in other words, a lossless step. The second main step is the quantization, which is omitted in lossless image compressors, since this is a non-reversible process. Next is the encoding step, often called entropy encoding. The two additional steps are the image source representation, and the compressed image data. The general block diagram for image compression is shown in Figure 2.5 below.

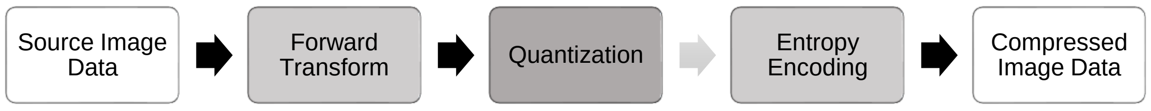

Figure 2.5 Building blocks for image compression

Different image compression schemes, utilizes different techniques, and may omit some of the steps. Some may not do anything with the *image source data* step, and code directly on the presented image, but some may require specific colour space representation or tiling before transformation. As there exist too many techniques in the different steps to explore all of them, some of the more common ones will be explained briefly in the subsequent chapters. The steps will be presented in the order they would appear, given that all of the building blocks (Figure 2.5) is included in the image coding scheme. The first step is the image source data, more specifically, how an image is produced, represented and pre-processed.

2.3 Image Source Representation

When creating a digital photography, an electro-optical sensor captures the light intensity from a light source (camera flash, sun or other light sources) [7, 45]. This sensor consists of small *photosites,* which is usually called pixels. The captured light intensity is then filtered through a colour filter arrays, depending on the wavelength. Normally this filter consists of three arrays[7], splitting the intensity into red, green and blue (RGB), as shown in Figure 2.6 below. The resulting RGB image, is a so-called *RAW* image. A RAW image is uncompressed, and very large, so these will be very demanding in processes for storage and/or transmission, and will therefore be needing compression. However, before the image is compressed, colour conversion is applied since the RGB isn't a good representation for humans to visualize different colours. In other words, proper perception of colours relies more on light intensity than single RGB colours.

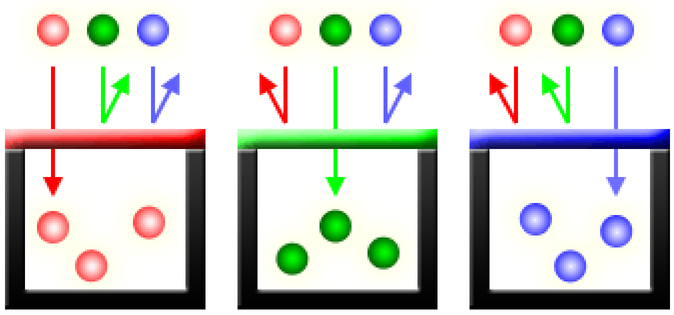

Figure 2.6 Illustration of RGB filters [46]

There are numerous methods for representing colours, depending on the different applications. For instance, computers often use RGB, printers use cyan, magenta and yellow (CMY), but YUV is common for human perception (see Subsection 2.3.1.2). All of these (among others) are based on three coordinates. However, these parameters do not tell which colour is displayed, only the amount of each component [47]. Some colour spaces are perceptually linear, meaning linear changes in values, will result in linear change in perception. This is not the case in many colour spaces used in computer graphics, on the other hand.

2.3.1 Colour Spaces

Since there are so many different colour spaces, it would be too much to go into detail about all of them. A few of the most common simplified spaces is still worth mentioning and will be briefly presented.

2.3.1.1 Red, Green, Blue (RGB)

As mentioned, RGB is often used intuitive from the uncompressed images. It is fairly easy to implement with the additive three colour system. It is one of the most used colour space, but it

[7] This is often a Bayer array, which contains twice as many green sensors as red or blue, due to the human eyes are more sensitive to green colour [46].

is non-linear with the visual perception and is therefore difficult for human description of colour [47, 48]. RGB is quite inefficient, since all of the parameters are highly redundant and correlated, meaning they all contain luminance information.

The RGB system is mostly used on displays and cameras, however when printing this is often changed to cyan, magenta and yellow (CMY) system, often with a black component in addition (CMYK). This is a simple linear transform from the RGB with the conversion shown in (1).

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

2.3.1.2 Luminance and chrominance (YUV)

The luminance (Y) and chrominance (U and V)[8] system removes some of the high correlation in the RGB system [7, 48]. Luminance represents the intensity of the light (or black and white), while chrominance is the colour components. The U refers to the difference between the red component and the luminance, while V is the difference between blue and luminance. This means these two categories (Y and U/V) can be treated separately, which is related to the human perception system. In lossy compression this separation is widely used to either completely remove the chrominance part or partly remove some of the colour (irrelevancy reduction).

Conversion between RGB and YUV are shown below in (2) [7, 49]. The transformation matrix can be different, since there are different versions of RGB (and YUV). The one shown below is a simplified version with base integers limited to 2, instead of floating points, which is more energy efficient.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 1/4 & 1/2 & 1/8 \\ 0 & -1/2 & 1/2 \\ 1/2 & -1/2 & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2}$$

2.3.1.3 YEF

This section explains the YEF colour space which is very similar to the YUV colour space[9]. In fact, the luminance component is very near-identical, but the chrominance components isn’t [50]. Both the luminance and chrominance components are uncorrelated, which means that larger quantization steps can be applied in the chrominance components (see Section 2.5), without affecting the overall reconstructed image quality. The main difference between YUV

[8] U and V is also known as Cr (red difference) and Cb (blue difference) in the JPEG format

[9] This applies mainly to the one described above, since different versions exists, which wouldn’t necessarily be near identical.

and YEF, is that the chrominance components are the difference between green and luminance (E) and the difference between blue and luminance (F).

YEF colour space was developed with endoscopic images in mind, since YUV is computationally expensive (when floating point numbers are used instead of the version described above) [51]. This results in energy savings due to efficient hardware implementation, since this colour space utilizes base integers of 2, which allows for simple bit shifting operations. The conversion equation is given in (3) below.

$$\begin{bmatrix} Y \\ E \\ F \end{bmatrix} = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/8 & -1/4 & 1/8 \\ 1/8 & 1/8 & -1/4 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{3}$$

2.3.2 Tiling

The term *tiling* is the process of splitting an image into small parts or partitions, which often consists of rectangular non-overlapping blocks [44, 52]. This is done because many embedded image processors may be unable to handle very large images, or it might have very limited memory available. Tiles reduces this problem, since the tiles can be handled independently, as if they were separate images and not just a partition of one image. Each tile is sent through all of the operations in the system; transformation, quantization and entropy coding (Figure 2.5). The same is the case for the decoder, which also will reconstruct the image, using the different partitions independently, reducing the memory requirement for the decoder. Tiling can be illustrated as shown in Figure 2.7 below, which shows three different levels of tiling. Firstly, the whole image is regarded as one tile, before it is divided into 4 tiles and 16 tiles.

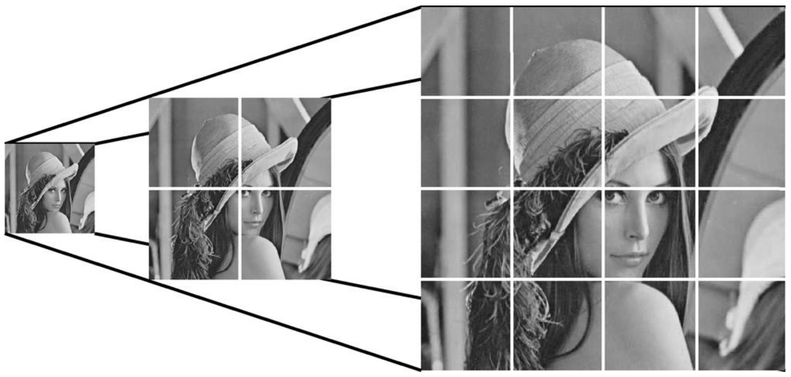

Figure 2.7 Three levels of a tiled pyramid for "Lena" image [53]

All of the different tiles in an image usually have the same size, but the tiles at the end of the image may be the exception [44]. If some tiles exceed the image size, these tiles are usually filled with zeroes in the surpassing area. As long as all of the tiles are the same dimension, the decoder will know how to reconstruct the image properly and efficiently. The size of the tiles still varies, and can be up to the size of the whole image (the whole image is regarded as one tile). The bigger the tiles, the better the reconstructed image quality, but it requires more memory and larger latencies in read write [52]. Smaller tiles are the opposite; they have smaller transfer time, require less memory but they have lower reconstructed image quality. A drawback with the smaller tiles, is that they can create what is known as tiling artefacts, which can be seen in reconstructed image as rectangular blocks[10]. On the other hand, tiles allows for some parts of the image to be compressed differently than other sections. This can result in variation in the image quality within the frame, also known as region-of-interest. These are factors which need to be considered when deciding tile size, in an image compression system. Regardless of what is chosen for the specific application, transformation coding will still have to be applied for efficient coding.

2.4 Transformation Coding

After the image source representation is satisfying according to the application, transformation is usually the next step in image compression. This is the process of transforming the image data into some other values (or another set). These "other values" can be interpreted as the image data is in another domain, which can be advantageous for identifying features that would be difficult to do in spatial domain. For example, some properties or characteristics can be easier to detect using frequency domain instead of the spatial domain. Some domains also provides a better foundation for the entropy encoding which comes later (see Section 2.6).

It is common to distinguish between two types of transformations; orthogonal block transforms and filterbank based transforms [54]. There are a variety of methods to transform within these two categories, and some don't completely fit into either of these two. Some of the common methods within these two categories will be briefly described; the discrete cosine transform (DCT) and subband coding (SBC). The DCT is used in the JPEG[11] standard, while a special case of SBC is used in the newer JPEG2000 standard. Another "transformation" which doesn't completely fit into one of those categories, but is quite common in digital signal processing will also be discussed, namely predictive coding.

[10] These tiling blocks aren't the same as the known blocking artefact, which happens with DCT transformation at low bitrates, because of high correlation between two blocks.

[11] Joint Photographic Experts Group (https://jpeg.org/)

2.4.1 Discrete Cosine Transform

The discrete cosine transformation (DCT) is a transformation which is related to the discrete Fourier transform (DFT). The main difference is that the DFT goes to infinity, the DCT is a finite set [55, 56]. Since it is a finite sequence, perfect reconstruction will not be possible, and it is therefore commonly used as a lossy compression scheme in image coders. Particularly, in the well-known JPEG standard. Note that DCT can still achieve near-lossless, but not completely lossless compression.

DCT takes an 8x8 greyscale block (tiling, Section 2.3.2), and compresses this block. When it is a colour image, the procedure is repeated for each colour channel, i.e. one colour channel is regarded the same as a grayscale channel. The values for the pixels are shifted at the entrance of the transformation, from [0, 2^P-1] to [-$2^{(P-1)}$, $2^{(P-1)}$-1], to ensure that the dynamic range is centred around zero. The centred 8x8 block is then transformed by (4) below. The result is that the 8x8 input block is transformed into a 64-point, 2-dimensional discrete signal. These 64 values are basically amplitudes, also known as DCT coefficients, based on the 64 input values [56].

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) \cdot \cos\left(\frac{(2x+1)u\pi}{16}\right) \cdot \cos\left(\frac{(2x+1)v\pi}{16}\right)\right] \quad (4)$$

Where:

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}}, & for\ u, v = 0 \\ 1, & otherwise \end{cases} \quad (5)$$

Inverse DCT (IDCT) is performed at the decoder side, given by (6) below, to get the 64 reconstructed pixel values based on the 64 DCT coefficients [56]. Mathematically, the DCT is a so called one-to-one mapping, between the frequency domain and the image. However, since it is a finite series, it can't be computed with perfect accuracy. Especially when the DCT coefficients are quantized, which is the case in the JPEG standard, i.e. in theory the DCT is a lossless transformation, but in practice this is near impossible to achieve so it results in lossy compression.

$$f(x,y) = \frac{1}{4}\left[\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v) \cdot \cos\left(\frac{(2x+1)u\pi}{16}\right) \cdot \cos\left(\frac{(2x+1)v\pi}{16}\right)\right] \quad (6)$$

2.4.2 Subband Coding

Filterbank transforms are in general based on frequency analysis of the signal by using filters [54, 57]. The general idea in subband coding (SBC), is to split the signal into different channels based on their frequencies, and process these independently. Splitting the signal is done by *n* parallel filtering (1 low-pass, *n*-2 band-pass and 1 high-pass) and down-sampling (decimated) by a factor of *k*. If the number of subbands is two, naturally there won't be any band-pass filters, and the channels will consist of low-band and high-band, as shown in Figure 2.8 below. The decimation is done since the output of the filters will contain more information than what is required, also known as *overcomplete output*[12]. If the decimation factor is equal to the number of parallel filters, the decimation is critical, meaning it is the limit which perfect reconstruction is guaranteed.

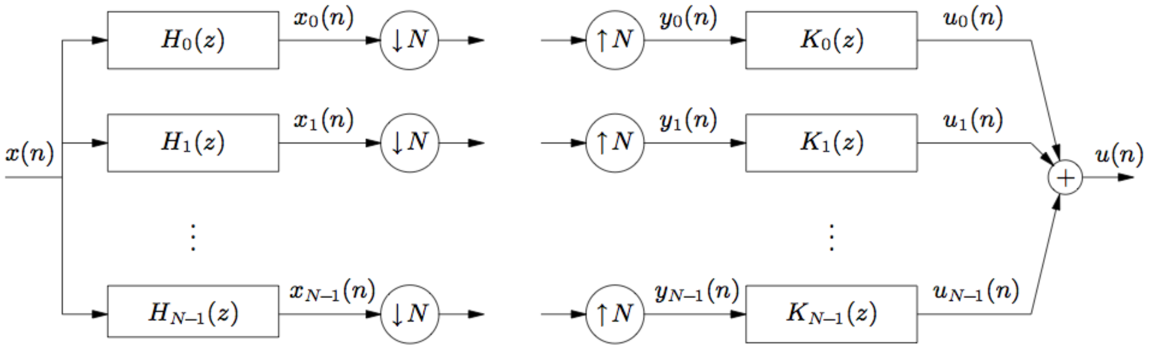

Figure 2.8 General subband coding scheme [53]

In Figure 2.8, both the encoding and the decoding part is shown. On the encoder side (left side), the process is often called the *analysis*, while at the decoder it's called the *synthesis [54, 58]*. Synthesis is the process of reconstructing the signal by interpolating the different subband signals, before filtering and combining using superposition of all the different frequencies channels. The filters depend on the kind of signal, which for images often is finite impulse response (FIR) filters. FIR filters gives linear transforms, and images are of *finite* size, making these kind of filters well suited for images. The design of the filters is critical to avoid aliasing[13] in the sub-sampling process, as the filters need to eliminate the alias components that occur when the interpolated signals are combined.

Until now, SBC has been mentioned as parallel channels, however there is no reason why all channels have to be in parallel. A special case of SBC is the cascade version, which can in a simple case be described as inserting another 2D coding scheme between the analysis and the synthesis part. If this is the case, the result would give three channels; low-low-band (LL), low-high-band (LH) and high-band (H), which could be expanded further [58]. This technique is

[12] In an *overcomplete* system, removal of some of the samples, won't make the system incomplete, but the result will still be complete.

[13] In aliasing, parts of the frequency spectre are copied, which will add to the signal (spectra overlap). This is an irreversible, unwanted process.

widely used in image processing, and it is what the basis for how the wavelet transform works, which is for instance used in JPEG2000.

2.4.2.1 Wavelet Transform

The wavelet transform (WT), or discrete wavelet transform (DWT), is a special case of the SBC. It utilizes 2D cascade scheme for decomposition of the image into, firstly, 4 channels; LL, LH, HL and HH. Images contain most information in the low-pass part, LL, so when compressing the image further, the cascade scheme is repeated on the LL part, and so on. This results in an image decomposition, as shown in Figure 2.9 below. The process is called *the dyadic decomposition*, which is a lossless transform, given correct implemented filters [44, 54]. If the image is tiled, the decomposition process is done independently on each tile.

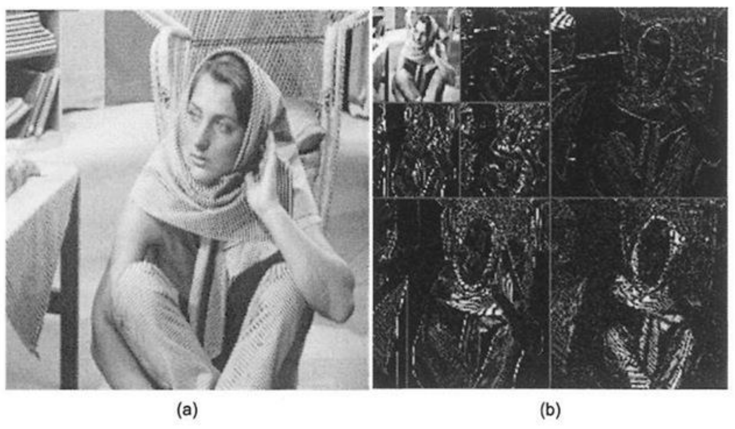

Figure 2.9 Two-level wavelet decomposition. (a) original Barbara image, (b) Resulting DWT decomposition image [59].

As in the more general case, the SBC, all of the different channels are down-sampled and filtered. The filters vary with the application[14], but the JPEG2000 standard has two main kind of filters, depending on irreversible or reversible transformation. Irreversible transform uses the Daubechies filter with 9-tap/7-tap (lossy) and the reversible uses same with 5-tap/3-tap filter (lossless) [44]. The Daubechies filters, are in fact one of the most common type of creating mother wavelets in WT [60]. Mother wavelet refers to the main function which is used in WT, and all other functions in the transform is scaled shifted versions of the mother wavelet. This

[14] The main requirement is that the generated function (wavelet) must be oscillatory, i.e. a *wave*.

property can be shown by looking at the discrete wavelet transform filters, where *h(n)* is the low-pass filter. The corresponding high-pass filter[15] will then be defined by (7).

$$g(n) = h(2N - 1 - n) \tag{7}$$

Equation (7) is valid for the one-dimensional case, but as for SBC, the two-dimensional case is just an expansion, or cascade version of the one-dimensional case. A common implementation of the cascade coupling is, for a *NxN* image *f(x,y)*, to first regard the image as a series of 1-D rows, and after as a 1-D series of columns [60]. The result from this one-level decomposition, is four sections of the image, which can be repeated again at the LL part, to get the two-level decomposition, as shown in Figure 2.9 above.

DWT gives some advantages in image processing. Since the image is divided into sections depending on the frequencies, de-noising, filtering and feature extraction are easily applicable, as well as compression. Most of these features involves eliminating or extracting some of the frequencies from the signal based on a threshold [60]. For instance, electromagnetic drifts can appear in wires during transmission, which will give high frequency noise. Such noise can be filtered or removed by the DWT domain, since most of the image information is located in the low-pass part of the image, and that particular noise is in the high-pass region. Direct compression gain from DWT is similar; by decomposition irrelevant information (often high-pass region) can easily be eliminated. This is a lossy compression method, but the indirect compression method doesn't need to be lossy. In the resulting other domain, values can be better for entropy coding than the original, which will result in a more efficient lossless compression than it would without the DWT.

Transformation of values into another domain, resulting in more efficient compression is common, but some methods don't require this transformation in order to be efficient. They can use other properties to obtain high compression ratio, which is the case for the predictive coding technique.

2.4.3 Predictive Coding

Predictive coding isn't directly defined as transform coding, since it doesn't transform the values into another domain, but it does change the values in the signal. The encoding scheme works by predicting the values, before subtracting the predicted from the original values. This creates an error signal, e(n), with the difference between these two signals. Prediction of the

[15] This is based on the quadrature mirror filter (QMF) algorithm, other variations exist, but the main principles are similar. Most of the popular discrete wavelets are in fact formed using QMF [60].

values is possible because many signals, especially images and speech/audio signals, have high correlation between consecutive samples. Since the resulting error signal has less variation than the original signal, the quantizer (covered in Section 2.5) can have a smaller decision region, resulting in a higher SNR [54, 61]. The general structure of the predictive coding system is shown in Figure 2.10 below. Omitting the quantizer part will give lossless coding.

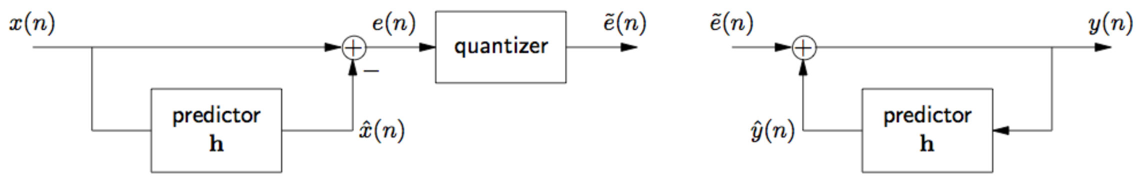

Figure 2.10 Quantized Predictive Coding System. Omitting the quantizer part will give a lossless system. Encoder on the left, and decoder on the right [53].

The system equations in this predictive system is:

$$e(n) = x(n) - \hat{x}(n) \tag{8}$$

$$q(n) = \tilde{e}(n) - e(n) \tag{9}$$

$$y(n) = \tilde{e}(n) + \hat{y}(n) \tag{10}$$

Where *x(n)* is the input signal, *e(n)* is the error signal, *q(n)* is quantizer error and *y(n)* is the output signal. In the lossless case, the quantization error, *q(n)* in (9), becomes zero which results to *y(n)=x(n)*. In the z-domain this translates into:

$$\tilde{E}(z) = X(z)\big(1 - H(z)\big) + Q(z) \tag{11}$$

$$Y(z) = \frac{\tilde{E}(z)}{1-H(z)} = X(z) + \frac{Q(z)}{1-H(z)} \tag{12}$$

The prediction error, *e(n),* will in this case be a continuous range of values, which isn't applicable in digital transmission. Because of this predictive coding in digital signal processing is often used with a quantizer, resulting in lossy compression [61]. It is fully possible to get near lossless compression using predictive coding, but this is often a bit more complicated and may include some transmission of side information [62].

2.4.3.1 Differential Pulse Code Modulation (DPCM)

Differential Pulse Code Modulation (DPCM) was first invented in 1950 by C. C. Cutler, and has become a very common technique in digital signal processing [63]. The system shown in Figure 2.10 is also known as an open-loop DPCM. In this system the encoder doesn't have any control on the reconstruction process, which will lead to the quantization error, *q(n)*. The additional error signal will then be amplified by the filter at the decoder, as shown in (12), which will lower the SNR [54, 61]. Because of this problem, the quantizer can be implemented inside the prediction loop, as shown in Figure 2.11 below. With the quantizer within the loop, the predictor always knows what the result is after the quantizer, i.e. *q(n)* is a part of *e(n)* and will therefore not be amplified by the filter in the decoder.

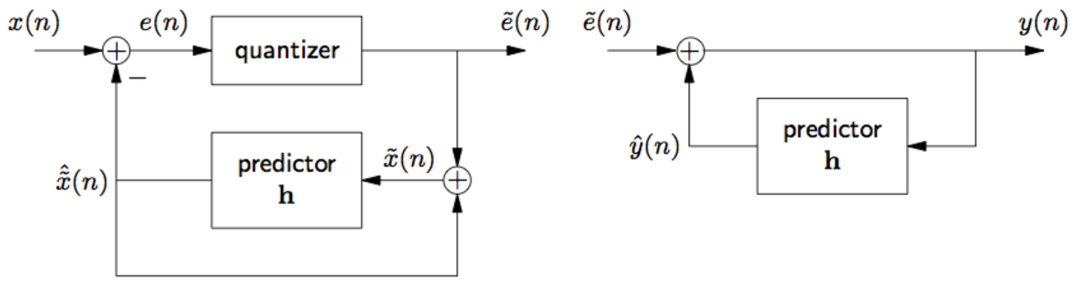

Figure 2.11 Block diagram of a typical closed loop DPCM system [53].

The system equations will in this case become:

$$q(n) = \tilde{e}(n) - e(n) \tag{13}$$

$$e(n) = x(n) - \hat{\tilde{x}}(n) \tag{14}$$

$$\tilde{x}(n) = \tilde{e}(n) + \hat{\tilde{x}}(n) \tag{15}$$

$$y(n) = \tilde{e}(n) + \hat{y}(n) \tag{16}$$

Which will be described in the z-domain as:

$$\tilde{E}(z) = X(z) - H(z)\tilde{X}(z) + Q(z) \tag{17}$$

$$\tilde{X}(z) = \big(Q(z) + E(z)\big) + \big(X(z) - E(z)\big) = X(z) + Q(z) \tag{18}$$

$$Y(z) = \frac{\tilde{E}(z)}{1-H(z)} = \frac{X(z)-H(z)\big(X(z)+Q(z)\big)+Q(z)}{1-H(z)} = X(z) + Q(z) = \tilde{X}(z) \tag{19}$$

As shown in (17)-(19) the problem regarding the quantization error being amplified is now eliminated. Since the encoder contains a full copy of the decoder, the encoder will always know what the resulting output signal will be [54]. Since the input signal on the predictor can’t be delayed, the quantizer has to be memoryless in the system. This will result in some penalty in SNR, with approximately 1.5dB, compared to optimal, but this isn’t considered a high price to pay in the overall system [61]. There are many different ways of design the quantizer which will affect the system performance (covered in Section 2.5), but the predictor also have a major role in the performance.

2.4.3.2 Predictor

The predictor in the DPCM system can be considered as the key component in the system since it highly influences the overall performance [54]. Naturally, the predictor will depend on the input signal, but a common technique is to model the input signal as a discrete vector. For an image, which is two-dimensional (2-D), one can either model as a series of vectors (multiple rows or columns) or implement directly multidimensional. One of the simpler models is to design the filters as a linear prediction of past samples, as shown in equation (20) and (21) (forward and backward prediction respectively)[64].

$$\hat{x}(n) = -\sum_{k=1}^{p} a_p(k)x(n-k) \tag{20}$$

$$\hat{x}(n-p) = -\sum_{k=0}^{p-1} b_p(k)x(n-k) \tag{21}$$

In this equation, p is the prediction order, a_p is called the prediction coefficients and the negative sign is for mathematical convenience and aligns with current practice. In the backward prediction, the prediction coefficients, b_p, is the complex conjugate of a_p but in reverse order, shown by (22).

$$b_p(k) = a_p^*(p-k) \tag{22}$$

A common method of finding the prediction coefficients is to model the source signal as an autoregressive (AR) process [54, 61]. The coefficients can then be found using the autocorrelation sequence by solving (23), which can be expressed with Yule-Walker equations given in (24) [64].

$$\gamma_{xx}(m) = \begin{cases} -\sum_{k=1}^{p} a_k \gamma_{xx}(m-k), & m > 0 \\ -\sum_{k=1}^{p} a_k \gamma_{xx}(m-k) + \sigma_w^2, & m = 0 \\ \gamma_{xx}^*(-m), & m < 0 \end{cases} \tag{23}$$

$$\begin{bmatrix} \gamma_{xx}(0) & \gamma_{xx}(-1) & \gamma_{xx}(-2) & \cdots & \gamma_{xx}(-p) \\ \gamma_{xx}(1) & \gamma_{xx}(0) & \gamma_{xx}(-1) & & \gamma_{xx}(-p+1) \\ \vdots & & & \ddots & \vdots \\ \gamma_{xx}(p) & \gamma_{xx}(p-1) & \gamma_{xx}(p-2) & \cdots & \gamma_{xx}(0) \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} \sigma_w^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{24}$$

Equation (23)-(24) shows that there is a linear relationship between the autocorrelation $\gamma_{xx}(m)$, and the prediction coefficients a_k. This linear relationship is not the case in the ARMA or MA[16] process which is nonlinear [64]. Linearity in the predictor can be advantageous for simplicity and work well in areas with high correlation. This is not the case for the edges in an image, so in images with a lot of edges, non-linear predictors have the advantages in terms of performance

[16] Autoregressive, Moving Average (ARMA) process & Moving Average (MA) process.

[54]. On the other hand, performance of linear predictors can be increased by having an adaptive predictor, giving adaptive DPCM system. Adapting the prediction filter depending on the input sample (or signal), can reduce the quantizer error, and achieve higher compression and higher SNR. The drawback with this system, is that it requires more computations (energy) from both the encoder and the decoder.

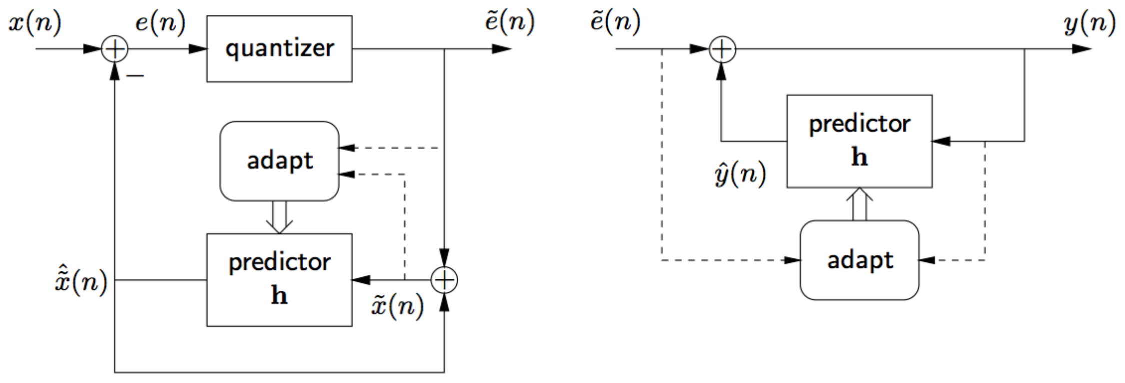

Figure 2.12 Adaptive DPCM system [53].

Since both the decoder and the encoder has access to identical signals, both can be made self-adaptive. In an AR model the predictor coefficients can be updated with each sample, i.e. no additional side-information has to be transferred, as Figure 2.12 shows. Switched prediction is another kind of adaptive prediction, which consists of different predefined prediction filters, switching between them depending on selected criteria [54]. A backward adaptive switched prediction system is what is used in the lossless and near-lossless coding scheme JPEG-LS.

2.5 Quantization

Quantization of a signal is in basic the process of rounding a signal value. Analogue signals have infinite possible values, so when digitalization of these values they are “rounded” to practical values. This rounding is also known as a many-to-one mapping, implying there will be loss in this process. However, in this analogue-to-digital converter (ADC) case, the quantization part is often of such high quality that it is considered “original” for all intents and purposes [37]. The quantization process within image encoding systems, will technically be the second time the image signal is quantized, resulting in lossy encoding. Omitting this is the lossless encoding, as mentioned, since it is related to the ADC (source) and not the image coder itself.

The general principle of the “second quantizer” (herby only referred to this one) is the same as in all quantizers. Quantizers round values into L different *quantization levels*, i.e. the number of outputs, which all lies within the *dynamic range* of the quantizer [37, 65]. Dynamic range is defined as the minimum and maximum values in quantizer which it can handle, i.e. the range of input signal values x_{max} - x_{min}. If the input signal exceeds these values they are either rounded (rounding) to these max- and min values, or the exceeding values can be discarded (truncation).

The allowed output values within the dynamic range are separated with distance Δ, also known as *quantization step size* or *resolution*. Each rounding in the quantizer, assigns each sample of *x(n)* to the nearest quantization level to produce the output $x_q(n)$. This results in some rounding error in the signal, also called *quantization error* or *quantization noise* ($e_q(n)$), which is limited to half of the resolution, as shown in (25).

$$-\frac{\Delta}{2} \le e_q(n) \le \frac{\Delta}{2} \tag{25}$$

$$e_q(n) \triangleq x(n) - x_q(n) \tag{26}$$

Many image encoders have implemented redundancy removal (Section 2.3 & 2.4) before the quantization part, since this can reduce the number of required quantization levels and/or the dynamic range. For instance, this reduction will be achieved in DPCM systems, which can reduce the produced quantization noise.

Equation (25) for the quantization error is, strictly speaking, actually valid for an *uniform scalar quantizer*. In this case, the step size, Δ, is a scalar constant (Figure 2.13 (a)), but this isn’t always the case. In *non-uniform quantizers*, the step size varies (Figure 2.13 (b)) depending on the assumed probability density function (pdf) of the input signal. In this case Δ is referred to as the scaling, but some still calls it the step size here as well, even though this isn’t strictly completely correct [37].

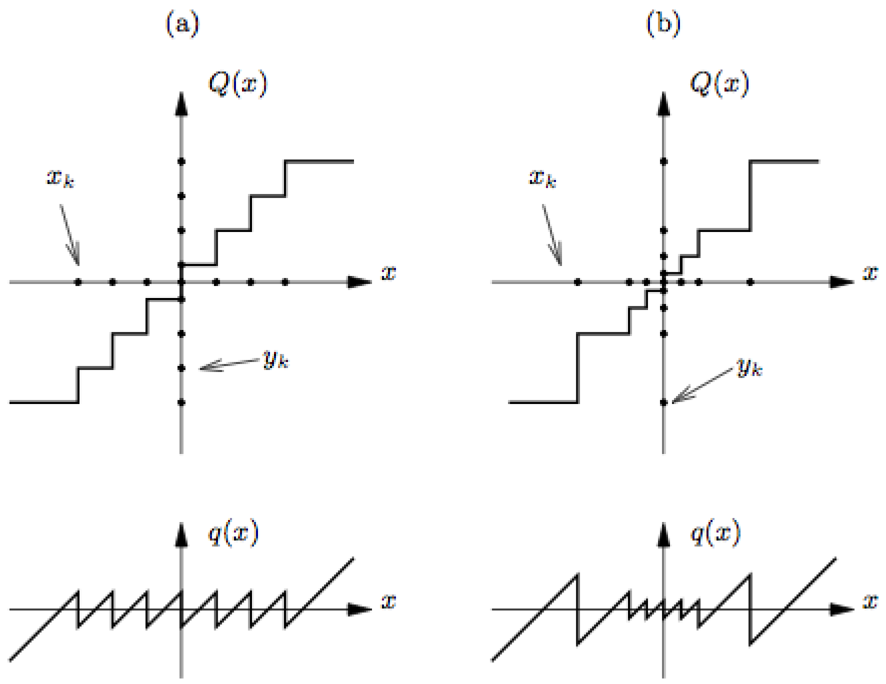

Figure 2.13 Midrise quantization Q(x) and quantization error q(x). (a) uniform and (b) non-uniform quantizer [53]. Note that $Q(x)=x_q(n)$ & $q(x)=e_q(n)$ in the text.

2.5.1 Uniform Scalar Quantizers

A uniform scalar quantizer is the simplest form, in terms of simplicity and implementation. It is called *uniform* because the step points on the horizontal axis is the same (uniform) as the output levels on the vertical axis [37]. If the quantizer has *b*-bits accuracy, this means that it has $L=2^b$ different output levels. The step size, Δ, is then simply defined as the dynamic range over the number of output levels (27), which will be the same for all the steps.

$$\Delta = \frac{x_{max} - x_{min}}{L} = \frac{x_{max} - x_{min}}{2^b} \tag{27}$$

Equation (25) will then be an absolute limit as long as the input signal doesn't exceed the dynamic range, if it does, the quantizer is said to be in *overload*. Within the dynamic range (also known as *granular region*) the performance is often represented by a *signal-to-quantization-noise ratio* (SQNR) or just SNR, described by (28) [37, 61].

$$SQNR[dB] = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_q^2}\right) \tag{28}$$

Where σ_x^2 is the variance of the input signal and σ_q^2 is the variance of the quantization noise, also known as mean squared quantization error. Assuming the quantizer isn't overloaded (i.e.

σ_x^2 small enough to ensure *$x \epsilon (-x_{max}, x_{max})$)*, and the input is uniformly distributed, the variance quantization noise becomes:

$$\sigma_q^2 = E\{e_q^2\} = \int_{-\infty}^{\infty} e_q^2 p_q(e_q)\, de_q = \int_{-\infty}^{\infty} \left(x(n) - x_q(n)\right)^2 p_x(x)dx \tag{29}$$

Where *$p_q(\cdot)$* and *$p_x(\cdot)$* is the pdf of e_q and x respectively. By approximation and a large enough L, the quantization error can be described as:

$$p_q(e_q) = \begin{cases} 1/\Delta, & |e_q| \le \Delta/2 \\ 0, & else \end{cases} \tag{30}$$

Giving:

$$\sigma_q^2 = \int_{-\Delta/2}^{\Delta/2} {e_q}^2 \frac{1}{\Delta} de_q = \frac{1}{\Delta}\left[\frac{{e_q}^3}{3}\right]_{-\Delta/2}^{\Delta/2} = \frac{\Delta^2}{12} \tag{31}$$

By inserting the step size, and having *b*-bits to represent each discrete output, we obtain:

$$\sigma_q^2 = \frac{\Delta^2}{12} = \frac{1}{12}\left(\frac{2x_{max}}{L}\right)^2 = \frac{1}{3}x_{max}^2 2^{-2b} \tag{32}$$

$$SQNR[dB] = 10\log_{10}\left(3\frac{\sigma_x^2}{x_{max}^2}2^{2b}\right) = 6.02b - 10\log_{10}\left(3\frac{x_{max}^2}{\sigma_x^2}\right) \tag{33}$$

The result in (33) implies that SQNR increases by 6dB for each bit added, i.e. 6dB increase for each doubling the number of quantization levels. This means that the quality of the output signal will also deteriorate at low- to very low bitrates, and will increase with high bitrates [66]. For the rest of the expression, it highly depends on the input signal. For instance, it can be shown that for uniformly distributed x, $x_{max}/\sigma_x=\sqrt{3}$, which means SQNR=6.02b, for a sinusoidal x, $x_{max}/\sigma_x=\sqrt{2}$ resulting in SQNR=6.02b+1.76 [61, 65]

Increasing the number of bits will give better quality, but will also cost more in terms of storage and transmission of images. Therefore, in lossy image coding, it is a common problem to try to minimize required bits without having too much distortion (which can translate into the quantization noise variance). When trying to minimize the required bits, this *mid-rise* quantization might be a disadvantage since it cannot have zero as an output value. *Mid-tread* (reference to a tread in a staircase) quantizers eliminates this by including the zero output value. However, it won’t be symmetric around the axis, which in turn can be a disadvantage if the input is modelled as a random variable.

In general, uniform quantizers are optimum as long as the input pdf is uniformly distributed, but if it isn't the uniform quantizer won't be optimum[17]. This is especially true if the bitrate isn't high enough [66, 67].

2.5.2 Non-uniform Quantizers

As mentioned, the uniform quantizers are advantageous in simplicity, and especially if there is no knowledge of the input signal, the average result can be better. However, when the pdf of the input signal is known or assumed, the case can be different. The steps can be chosen independently depending on how the signal is in order to minimize quantization error function [37]. This will reduce the quantization error in some regions, but can increase in some others, as compared with uniform in Figure 2.13. However, finding the different steps sizes for different regions can be a complex approach. This can be solved using numerically computation, which was done by Lloyd-Max (resulting in the Lloyd-Max quantizers) for Gaussian distributions. Because of this, effective non-uniform quantizers can be difficult to design without knowledge of the source statistics [68]. For some applications, some statistics can be exploited. For instance, speech signals have high probabilities for low amplitudes, which can be coded as a non-uniform quantizer. Another good solution is simply to transform the speech signal into something that looks uniform and then use a uniform quantizer (compress the larger amplitudes in the speech signal).

For image coders (and video sequence coders) it isn't necessarily the same case as speech coders, but the same technique is used in encoders which can have high bitrate. At low bitrates it is common to use a special case of the non-uniform quantizers, namely the *dead-zone* (also known as dead-band) quantizer.

2.5.2.1 Dead-zone Quantizer

The Dead-zone quantizer is a non-uniform[18], mid-tread quantizer, basically by having a larger region (or step) around zero. This means that more of the input values will be rounded to zero, which reduces the required rate [7, 69]. This property also results in an increase in the quantization error around this area, but the potential savings in rate if many samples are near

[17] In terms of the quantizer itself, for some specific applications uniform can be optimum in terms of simplicity and complexity.

[18] Some do regard dead-zone quantization as a uniform, since it can often be uniform in all regions, except for the region around zero.

zero value can compensate for this increase in e_q. Dead-zone quantization has been proven quite effective in image compression, and is therefore used in Part 1 in the JPEG2000 standard [44].

There are different versions of the dead-zone quantizer depending on the source. For instance, the optimal Entropy Constrained Scalar Quantizer (opt-ECSQ) is used for sources having exponential and Laplacian probabilities density function [69]. On the other hand, this has been proven to be quite complex in designing and implementing. The Uniform Reconstruction with Unity Ratio Quantizer (URURQ) is an improved version of the Uniform Threshold Quantizer (UTQ), which can be used as a very well approximation of the ECSQ. URURQ has a wide zero region, but has equal uniform steps outside this zero area with step size Δ. The dead-zone region in the middle is a function of this step size, given by (34), as well as an offset value given by (35).

$$DZ_{region} = 2\big(\Delta - \delta(\Delta)\big) \tag{34}$$

$$\delta(\Delta) = 1 - \frac{\Delta e^{-\Delta}}{(1-e^{-\Delta})} \tag{35}$$

The total URURQ and its inverse can then be described by (36)-(37) below.

$$x_q(\Delta,\delta) = sign(x(i,j)) \cdot \max\left\{0, \left\lfloor\frac{|x(i,j)|+\delta}{\Delta}\right\rfloor\right\} \tag{36}$$

$$x_{rec}(\Delta) = \Delta \cdot x_q(i,j) \tag{37}$$

Where *(i,j)* describe the location of pixel values for an image, and x_{rec} is the reconstructed value at the decoder. Comparing URURQ to the more complex algorithm opt-ECSQ, the difference in rate-distortion performance (SQNR) for different bitrates is at a maximum 0,0021dB [69]. Meaning that since the URURQ is a lot simpler, it would be advantageous to use that quantizer at lower rates. Note that for high bitrates the uniform will generally still be optimum compared to any dead-zone quantizers [66].

2.5.3 Adaptive Quantization

Since many application has to compensate for different input signals (e.g. different speech variances or different kind of images), problems can occur in the quantizer part for the encoder (overload or mismatch in the steps). One well used solution to this problem is to adapt the quantizer step size, Δ, and/or the dynamic range of the quantizer [37]. The general idea of this is that the encoder will adapt to the input signal based on some local statistical properties (pdf's) of the signal. It is common to distinguish between two kinds of adaptive quantizers; Forward adaptive (FA) and backward adaptive (BA) quantizers. Both of these methods have advantages and disadvantages.

Forward adaptive quantizers extract the step size from the input signal of the quantizer [37]. Calculation of Δ is done over a block of data, and is therefore not changed for every input value. Because of this block, a buffer is needed at the encoder which introduces a delay in the signal. After the calculation of the step size, it is applied to the quantization, however, the same step size is required at the decoder. In other words, transmission of side information is required in FA quantizers. Naturally, the size of the blocks affects the amount of side information needed to be transmitted, since smaller blocks result in more calculations of Δ (and larger delays) [70]. Performance of the quantizer will also increase with smaller the blocks, so this implies a trade-off between quality and delay/transmission rate.

Backward adaptive quantizers extract the step size from the output signal only of the quantizer, and applies Δ to the next sample. Since this doesn't require the input signal, the calculation can be done in both the decoder and the encoder. Therefore, there is no need to transmit side information and no delay, which is an advantage over the FA quantizers [37, 70]. BA quantizers operate on a sample-by-sample basis, and not over a block, which also eliminates the need for a buffer. However, since it is the output signal that is used for calculation, the quantization noise is also used in determining the parameters, resulting in a loss in performance. This also increases the sensitivity to errors in the quantizer, which can be difficult at very low bitrates.

A third alternative to adaptive quantization can be switched quantization, or quantization banks, similar to switched predictors (Section 2.4.3.2) [67]. This eliminates the need to calculate the different parameters, but requires more storage of the different quantizers alternatives, and it doesn't eliminate a buffer or the need of statistical analysis of the input signal.

2.6 Entropy Encoding

Entropy coding[19] is the last main building block in image compression (Section 2.2.2). The main principle of entropy coding is to find new ways of representing the information based on statistical properties. This is a lossless compression method based on redundancy reduction. When coding an image (or any kind of source) it is represented by bits, and the "standard" way is to use equal amount of bits per source letter, or pixel [8, 37]. Thus, if a source consists of 2^3=8 different symbols, at least 3 bits is required in order to represent all of the different symbols. This is known as *fixed-length code,* or *block code.* Every symbol would then have *unique decipherability.* In other words, the decoder would always know which symbol it is supposed to decode. However, this is an ineffective coding method, since more bits is used than

[19] Note that variation of this term exist, some simply call this "coding", and some include "channel coding".

what is needed. Different source letters have different frequency appearance, or different probability of appearing. Comparatively, if the source was an English text (including space), and a random character was selected, there is a 10.1% chance it's an "E" and a 0.1% chance of "Q" [71]. Naturally, if "E" was coded with 1 bit, and "Q" with 4 bits, the 4 bit would appear less, while the 1 bit would be more frequent. By assigning different bit length to different symbols, based on the probability of appearance, the total average code-symbols per source-character decreases. This is also known as *variable-length coding* (VL code).

Different versions of VL codes exists, but an important feature of the code is that it has unique decipherability, so the decoder can't misinterpret a symbol. Common techniques for VL coding can for example be arithmetic coding, Huffman coding or run-length coding.

2.6.1 Arithmetic Coding

In arithmetic coding the whole message is coded into a string, ranging from 0 to 1. Each symbol in the message is then represented as a fraction interval on this "main" interval, with a size relative to the symbol probability [72]. Successive symbols are then coded in a recursive manner in this interval. Symbols with high occurrence probability is coded with few bits, while low probability occurrence is coded with more. This is best illustrated with an example. Imagine an alphabet consisting of six symbols (a, e, i, o, u, !) with probabilities and rages shown in Table 2.1. The message that the encoder is sending is "*eaii!*".

Symbol	Probability	Range
a	0.2	[0, 0.2)
e	0.3	[0.2, 0.5)
i	0.1	[0.5, 0.6)
o	0.2	[0.6, 0.8)
u	0.1	[0.8, 0.9)
!	0.1	[0.9, 1.0)

Table 2.1 Arithmetic coding example [72]

The decoder, which know the range is [0,1) will then start with the first symbol, and decode it as "e" since the received value lies between 0.2 and 0.5. This new interval is then examined where it decodes an "a" since it lies between 0 and 0.2. The process is then repeated as shown in Figure 2.14.

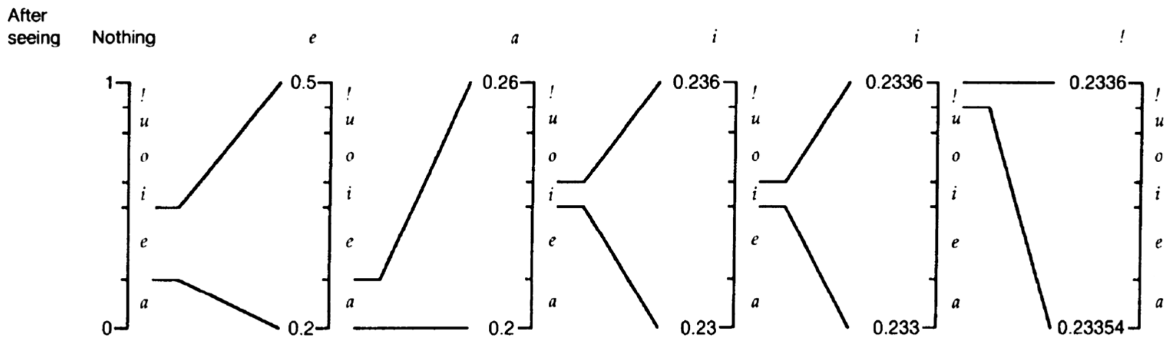

Figure 2.14 Arithmetic coding example [72]

This recursive model does ensure as the whole message can be encoded as a single number. However, the longer the message, and the bigger the alphabet, the more bits are needed in the representation. There is also a potential problem with unique decipherability in implementing it in practice, since "0" could represent a, aa, aaa etc. Some solutions have been suggested by inserting comma (or other symbol) or transmission of the size [72].

2.6.2 Huffman Coding

Huffman coding is another form of entropy coding in order to code the coefficients. Basically, the encoding algorithm works by assigning every source symbol a sequence of bits, approximately equal in length to the information that symbol contains [37, 73]. The amount of information[20] in a symbol is given by the frequency (or probability) of that symbol. This ensures that the most frequent symbols have the least amount of bits, while symbols of low probability have longer code. Subsequently, the average code-word length will be reduced and approaches the fundamental limit set by the entropy. Essentially, the algorithm generates the code by replacing the prescribed set of source statistics with a simpler one. This is a step-by-step process, which works in the following way:

1. The source symbols are listed in order of their probability (or frequency), and the two least frequent symbols are assigned 0 and 1.
2. The two source symbols with the lowest probability is then combined into a new symbol with probability equal to the sum of these independent symbols. This new symbol is then placed in the sorted list with its new probability.
3. The process is repeated until only two symbols remains which are assigned 0 and 1.

[20] As probably known, the less likely a symbol is to appear the more *information* that symbol contains. If a symbol has high probability of occurrence it contains less information.

By working backwards through the list, the different code words are found, and each symbol can be assigned these code words. However, this isn’t a unique process as there isn’t any ruling for which symbol is assigned 0, and which is assigned 1. If more than two symbols have equal probability, there are more than one way to decide which two symbols are combined. This doesn’t affect the average code length though. Note that this uniqueness in the process isn’t to be confused with unique decipherability which the code still has, given equal ciphering tables at the encoder and the decoder. Huffman coding is a quite efficient encoding process, but it has the disadvantage of requiring storage for lookup tables. Some work has been done in order to make Huffman coding more energy efficient by reducing the required switching activity[21], but it doesn’t eliminate the need for tables and can require some more pre-computations [74].

2.6.3 Run-Length Encoding

Run-Length encoding (RLE) is a relative simple form of lossless entropy encoding[22] of the source symbols [54]. The technique is one of the most-widely used, and was used in the early television compression scheme in 1967 [75]. RLE is basically compression by grouping, meaning that consecutive data is grouped into a single value and then counted. If many consecutive samples are equal, the amount of these values will be transmitted instead of all of them. This package is stored in a “RUN/LEVEL” fashion, often with an End-of-Block (EOB) at the end. For example, if the sequence “aaaaabbbcccccd” is to be transmitted, this will be coded as “(5)a(3)b(5)c(1)d”. Naturally, this will be quite effective if many samples have the same value, but will be ineffective if there is a very high variation in the samples. Since images have high correlation between sample values, RLE is a very common image coding technique, and is used in for example, JPEG, MPEG[23] and H.26x [76]. Note that RLE can be made “lossy” if a threshold is included in the encoding. This can be regarded as a simple form of rounding (quantization). If the difference in consecutive values are below a given threshold it is regarded as the same sample value.

Since RLE is a 1D coding scheme, it is necessary to unwrap the 2D image, i.e. read the image as a single data stream. Different methods exist on reading the image as 1D, and the way it is read, is defined as the *scan sequence*. The most common scan sequence is the *zigzag* scan,

[21] Switching activity is the process of changing a value. For instance, encoding “00” requires none switching activities, while “01” requires 1 switching activity.

[22] Note that some doesn’t define RLE as “entropy encoding”, but is regarded as a pre-stage before the entropy coding (for instance, RLE first then Huffman coding after).

[23] Movie Picture Expert Group (http://mpeg.chiariglione.org/)

where the samples are read in a diagonal order. Other simple scan sequences can be horizontal or vertical scans, but more complex sequences exist and might be application (or type of image) depended [54]. For instance, if the image contains a lot of vertical lines (city image) the horizontal (row-wise) scan is optimal, but generally, the zigzag scan is the most advantageous.

Naturally, RLE is quite favourable in very low complexity encoders, since it doesn’t require any buffers and has very low implementation complexity. However, extra bits are required in RLE for the sign of the nonzero indices. So for very low complexity coders, a special case or improvement of the RLE was developed by some researchers at the University of California, named the Stack-Run encoding [77].

2.6.3.1 Stack-Run Encoding

Stack-Run (SR) encoding works by partitioning the (quantized) coefficients into two subgroups, where one group contains the nonzero coefficients while the other contains the zero valued coefficients [77, 78]. The groups are then represented as a stack or a column in binary notation with the most significant bit (MSB) on top, and the least significant bit (LSB) on the bottom. The stacks are then mapped into a symbol stream using a special alphabet.

SR coding is composed of a four symbol alphabet (0, 1, +, -), where “0” and “1” represent run-length of nonzero coefficients (levels) and the symbols “+/-” is used for the MSB of the nonzero coefficients and the successive zero coefficients (runs). This is because if only the “0/1” is used, it won’t be possible to distinguish between nonzero values and runs of zero values without more context. Since all binary run lengths start with 1, one can omit the MSB from most of the run-length representations without loss in information. Potentially, this can be a problem if the run has a length of one, which would not be representable if all MSB symbols are eliminated. Therefore, the MSB “+” symbol is only retained when the runs are of length 2^k-1, where k is an integer.

When the resulting bit stream changes from “1/0” to “+/-” or vice versa, the decoder know a new code word has started since all code words starts with either “+” or “-” (MSB). Since all words starts with indication of MSB, an EOB isn’t required in this method.

Further entropy encoding (such as Huffman or Arithmetic) is sometimes applied after the conversion to SR since it can be quite affected by channel noise. Some work has been done to minimize the need for additional entropy coding. This was done by making some assumptions of the channel (memoryless, binary symmetrical channel), and code each of the subband independently [79]. This results in performance gain, but also a slightly increase in complexity.

2.6.3.2 Bit-Plane Encoding

Bit-plane encoding utilizes separation into the different bit planes in order to achieve effective lossless compression together with for example RLE [54, 80]. The basic principle of bit-plane coding can be easily imagined if one considers an 8bit grayscale image (unit8), by using an AND-operation for each of the 8 planes (for instance, ANDing with 10000000 gives the 7th bit-plane, ANDing with 01000000 gives 6th plane). Each of these planes will then contain less information than the whole image, which will result in more zeros. RLE encoding can then effectively reduce the required transmission rate [81]. Every coded plane is ended by an End-of-Plane (EOP) block, indicating the start of the next plane.

This separation technique is lossless, but can be made lossy by eliminating one of the planes which can remove the need for a quantization part in the image coder. When a specific bitrate is required, the encoding process can stop anywhere, while the image is still decodable. This technique can be demanding since a buffer is required at very low complexity applications, due to each plane needs to be treated separately.

3 Wireless Capsule Endoscopy

The Wireless Capsule Endoscopy (WCE) have, in contrast to the other kinds of endoscopic equipment, relatively poor image quality. This is a direct result of the amount of power available, since WCE have a small battery, while the other methods are connected to an external power supply. When the equipment has "unlimited" power available, they can use high bandwidth, higher bitrate and complex image compression algorithms. Capsule endoscopy on the other hand, which, as mentioned, is just a small pill, aren't connected to an external power supply. It only gets the power it needs from a small battery, and the battery need to last until the journey through a patient's body is complete (approximately 8 hours). The image quality is therefore significantly poorer with the capsule endoscopy, compared to the other endoscopic methods.

Since the image quality is poorer, capsule endoscopy is mostly used where the other methods can't reach (small intestine), and the other methods are preferred otherwise. However, this is not the case for the patients. Capsule endoscopy isn't felt by the patients, while the other methods are known to give patients discomfort, and in worse cases scratch the wall inside the GI tract so the patients start to bleed [6, 7]. This isn't desirable, especially if the reason for the test was to find another bleeding inside the GI tract. So it would be advantageous to improve the image quality of the capsule endoscopy, to minimize the patient discomfort, while the physicians are able to diagnose. This chapter is therefore dedicated to explore how the capsule work in details, before the theory can be applied in order to improve the image quality.

3.1 The Sender and Encoder

The WCE system consists of two main parts; the sender and receiver [82] (receiver will be covered in Section 3.3). The sender is the small pill, which is about 26 x 11mm in size and a weight at about 3 gram [83]. The capsule is swallowed and as moving through the GI tract by the peristalsis, the same way as the food (Section 2.1.1). At the front of the capsule, a small camera lens (with RGB filters) and six LED light source, captures the frames at 2-4 frames per second (fps). The frames are thereafter processed by a microchip which codes the images before they are transmitted to the antenna in the back which sends the signal through the body to the receiver. All of the electrical components are powered by a small battery in the middle, which is required to be able to power the whole system for approximately 8 hours. Illustration of the capsule is shown in Figure 3.1, below. To protect the electronic components, they are encapsulated by a biocompatible plastic, which can resist all of the different digestive juices.

The capsule is made for one-time use, and after it has passed through the patient's body it exits the body together with the stool, ending in the toilet.

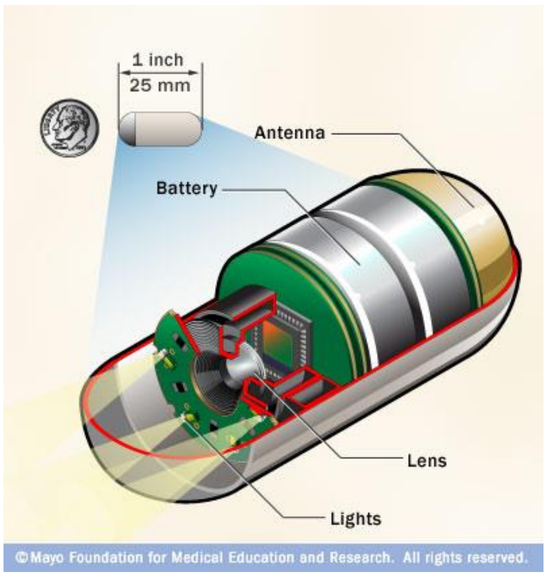
1 inch
25 mm
Antenna
Battery
Lens
Lights
©Mayo Foundation for Medical Education and Research. All rights reserved.

Figure 3.1 Illustration of the WCE [82].

As mentioned, the image capturing sensor captures the raw images, which is of very high quality. These are then compressed in a lossy format, before transmitted. Basically there are two main ways of improving the image quality of the capsule; increasing the amount of available power or improving the algorithm. Since the capsule is required to be very small (compact), in order for the patients to swallow it, there is not very much to room for hardware changes. Some research has been conducted in order to try wireless transmission of energy to the capsule [11, 12]. However, this is still at an early research stage and will need further development before it can be utilized in practice. Therefore, the most effective way to improve the image quality, may be to improve the algorithm.

Different versions of the WCE exists, by different companies with different trade secrets. This means that a full system description is very hard to obtain. The focused capsule is the standard version, PillCam™ SB from Given Imaging Ltd. which is one of the most widely used, and the most common design. Unfortunately, information about the specific algorithms used in this pill isn't available, so the described algorithm is based on the proposed algorithm was developed by researchers from Oslo University Hospital and Norwegian University of Science and Technology [7].

3.1.1 The Image Compression Algorithm

The image compression algorithm in the capsule has to consume a minimal amount of power, in order to achieve highest possible image quality throughout the whole journey through the GI tract. For this reason, every part of the algorithm is carefully selected to maximize the image quality, while minimizing the required physical size of the microchip and power consumption. This is done by five main components in the coding scheme; colour transformation, multi-rate, DPCM coder, dead-zone quantizer and a stack-run entropy coder, shown in Figure 3.2 below.

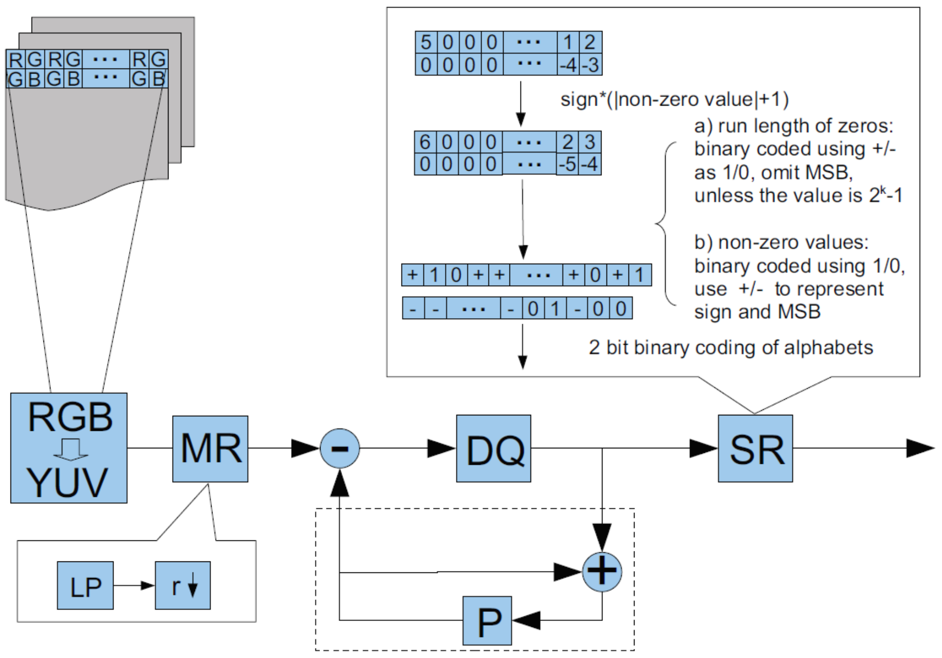
sign*(|non-zero value|+1)
a) run length of zeros: binary coded using +/- as 1/0, omit MSB, unless the value is 2^k-1
b) non-zero values: binary coded using 1/0, use +/- to represent sign and MSB
2 bit binary coding of alphabets
RGB YUV
MR
DQ
SR
LP
r↓
P

Figure 3.2 Block diagram of the algorithm [7].

3.1.1.1 Source Representation

As each of the RAW image frames enter the encoding scheme, the colour space is converted from RGB into the YUV-space. Each of the different colour components are treated independently throughout the algorithm. The conversion process is done by the simplified transformation matrix given by (2) (Section 2.3.1.2), which is more attractive towards binary treatment [49]. Every image frame is coded independently, so the whole system is regarded as an image coder, and not as a direct video coder. The avoidance of floating points in the transformation allows for single bit shifting operations in the colour conversion, which is very energy efficient.

The images aren’t directly tiled (Section 2.3.2), except from separation of the colour components. This choice would initially seem as it would require more memory and introduce more latency, but by utilizing the characteristics of the camera sensor, it don’t. Since endoscopic images have little variance in the chrominance (U, V) components compared to the luminance (Y), they are treated different in the DPCM [7]. In the prediction, the chrominance components are only predicted by the past sample on the same row. In other words, for these two colour components, the whole row can be regarded as a tile, giving the number of tiles equal to the number of rows. This isn't the case for the luminance component, so looking at the whole frame, no tiling is applied.

3.1.1.2 Multirate

Multirate (MR) can be regarded as a special case of one dimensional subband coding (Section 2.4.2). The MR system consists of two main parts; the decimation and the interpolation [84]. In the decimation part, the goal is to decrease the sampling rate of the signal, while the opposite is the goal in the interpolation. As in SBC, a filter is following the sampling rate compressor and expander, which here is called a digital anti-aliasing filter (decimation) and anti-imaging filter (interpolation). In contrast to SBC, the signal isn't divided into different frequency parts (high part and low part), but the whole signal is regarded as a single part. This sample rate conversion allows for increased computational efficiency as well as improved performance, and it is widely used within sound- and image processing.

In the WCE, the MR part is used to remove some part of the spectrum by using a simple low-pass (LP) filter instead of the anti-aliasing filter. This process can be applied since at very low bitrates will it be redundant to use bits to describe the spectrum which is below allowed reconstruction noise. The down-sampling is performed to ensure that the low-passed signal is used in the whole full-band available. The down-sampled signal is then encoded, and the interpolation part (up-sampling) is done at the decoder to save required computation and transmission in the sender.

This MR process is performed in both the rows and the columns, which means a reduction of r^2- source samples. This also means that $1/r^2$ bits are required compared to the original signal. This is a lossy process ($r \neq N$) in terms that perfect reconstruction isn't guaranteed. Generally, MR can be near-lossless depending on the filters (not the case with simple LP filters).

3.1.1.3 DPCM Predictive Coding

The algorithm in the WCE uses the time-domain source coding technique DPCM (described in Chapter 2.4.3.1). It utilizes the closed-loop (backwards) DPCM, which ensures that the quantization noise problem is eliminated. The quantizer used in this predictive system is the dead-zone quantizer, described in Section 2.5.2.1, in which the corresponding quantization equations is given by (34)-(37).

The prediction of pixel *(i, j)*, within the DPCM system is based on the reconstructed pixel values from neighbouring pixels, above *(i-1, j)*, to the left *(i, j-1)* and diagonally above *(i-1, j-1)*:

$$\hat{\tilde{x}}(i,j) = a\tilde{x}(i-1, j-1) + b\tilde{x}(i-1, j) + c\tilde{x}(i, j-1) \tag{38}$$

Where a, b, c are the prediction coefficients, which in this case are averages with relationship in (39). The "a" can be changed in the interval [0.65~0.95], but a value around 0.8-0.9 gives a good prediction.

$$\begin{gathered} a = 0.8 \\ b = a \\ c = -a^2 \end{gathered} \tag{39}$$

Note that equation (38) is for the luminance component, while the two chrominance components only use one reference pixel, and therefore only uses the prediction coefficient "a". This can be done since endoscopic images of the GI tract tend to have very little variation in colour [7]. In addition, it is advantageous in terms of memory requirements to only use the rows since CMOS image sensor captures images row-by-row [85]. By only using prediction from the same row, there is no need to temporary store other rows.

3.1.1.4 Stack-Run Encoder

In order to avoid storing of lookup tables, buffers and potentially computations, the algorithm uses the effective Stack-Run (SR) encoder instead of, for instance, Huffman or Arithmetic coding (see Section 2.6). The SR encoder has been proven quite robust against bit errors, and makes additional entropy coding redundant [79]. SR encoding is especially beneficial due to the dead-zone quantizer in the DPCM part. The quantizer forces more of the error signal towards zero, which increases the successive numbers of zeros, and the effectiveness of the RLE system.

Since the WCE has very limited implementation space and limited power, additional entropy coding is omitted. The four symbol alphabet in the SR coding are therefore chosen to be composed by a simple 2-bit representation.

After the image signal is SR encoded it is transmitted over the channel to the decoder. Since the image coding algorithm is the focus, channel coding is omitted from the simulation, as this won't affect the resulting image encoder/decoder. However, noise from the channel will affect the received image quality, by introducing bit-errors. Therefore, the channel will briefly be described next, to explain how the whole system will work, and what to take into account when implementing in a real system.

3.2 Transmission

When the frames are coded, they are instantly sent to the decoder through the human body. This is done to avoid the need for storage in the WCE. The transmission is, in this case, done by a radio transmitter. The antenna is sending out the signal with a frequency at 434,1 MHz and with a bandwidth at 1,6 MHz [83]. Some research has been done in this area and ultra wideband (UWB) is expected to change this in the future.

Ultra wideband (UWB) technology for short distance transmission, is a very energy efficient technology. UWB has a bandwidth from at least 500 MHz and up to 7,5 GHz, i.e. very high data rates can be achieved [10]. The high rates will mean that higher quality, resolution and/or frame rate can easily be obtained. In other words, lossless image compression can be implemented cheaply. This isn’t only due to the high data rates, but also because of the Shannon-Hartley theorem. The theorem states that parts of the wideband can be spent for energy transfer back to the capsule, which would be very advantageous in ultralow power and low SNR devices [9]. One of the drawbacks with this technology is the receiver complexity. Since UWB uses so low power it is hard to detect, and therefore the receiver requires very power hungry components, such as very high speed A/D converters and high-gain, low-noise amplifiers. In addition, it requires very extensive signal processing at the receiver, which makes the receiver unsuited to be powered by a battery, and would require the patients to be connected to an external power supply [86]. The technology is very promising, but will still need more research before it can be used in practice, so for now, the concentration is on narrowband radio technology (used in current system).

3.2.1 The Wireless Radio System

In a wide perspective, wireless systems is defined as a system which allows for the communication of information between two points without the use of wired connection [87]. This communication is possible because electromagnetic (EM) waves can propagate trough air and matter without conductors. Propagation through matter and reflection from different surfaces will result in fading of the signal, which is one of the main challenges in wireless systems. These EM waves are produced by antennas which convert a guided EM wave on transmission line and into a plane wave. Antennas are inherently bidirectional, in that all can be used for both transmission and receiving of EM waves [88].

Noise is one of the other main challenges in wireless systems. Sources for noise exists everywhere, from the sun, atmospheric, other wireless systems, thermal noise etc. [89]. The term “noise” is often used to describe any unwanted signals that tends to disturb transmission and processing of signals, in an uncontrollable matter. In communication systems is it common to calculate with white noise, which contains all frequencies. This is impossible in practice, because if a signal contained all frequencies, it would require unlimited power. White noise still has an important feature in the statistical mathematics, and is used for analysis of worst case scenario. If the noise is too severe, increase in signal power could be a solution in achieving acceptable SNR depending on the application.

However, there is a limit to how much one can increase the emitted power from the antennas. Increase in power would require more energy, and can be dangerous to the human body [87]. The energy in the waves will be converted into heat when exposed to too much radiation from EM waves, the same way a microwave heats food. This can be especially dangerous for some organs in the body at close range. IEEE has researched the dangers of human exposures to EM waves [90]. As can be seen from Figure 3.3, at frequency of around 400 MHz, recommended maximum exposure is at 3W/m^2 (3mW/cm^2). In comparison, the WCE transmits at 434 MHz, and effective radiated power (ERP) is at 57nW [83], which is well below maximum recommended exposure.

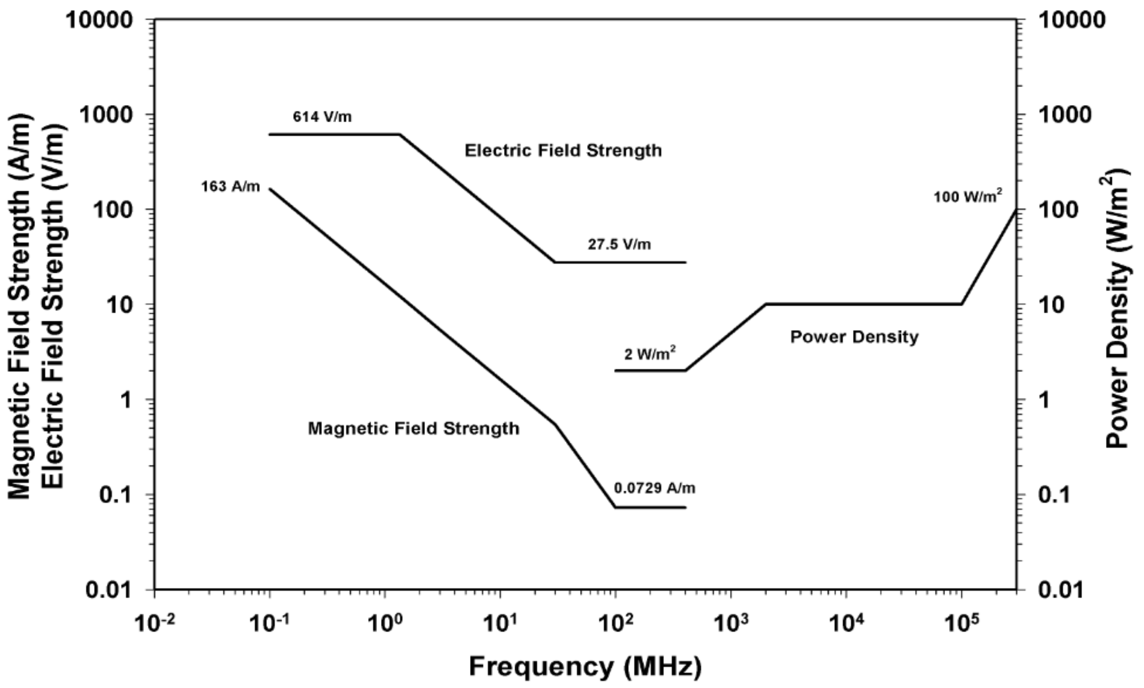

Figure 3.3 IEEE standard on human exposure to EM fields [90].

3.2.2 The Human Body as a Communication Channel

Describing a communication channel precisely is impossible, as there are too many random variables (all reflections, diffractions, scattering, penetrations etc.) [91, 92]. Instead it is common to use statistical model to determine the probability that channel parameters attain certain values. This is especially true when the sender and/or receiver are moving with variation in surrounding environment. All of the different noises and channel parameters will directly impact the received SNR, by interfering with the transmitted signal. As the distance between the sender and receiver gets larger, the different EM waves characteristics will be more influential. However, if the waves are passing through mediums or the transmitted effect is very low, these parameters will have impact on the received SNR.

In the case of WCE, the wireless communication channel will be the human body. Since every person have different anatomy, the channel will differ. The largest variation in channel parameters in this case is around the skin (closest to the receiver), and this will affect how the radio signal propagates. This is because the human skin consists of a mixture of dead and alive cells, together with fat and sweat [93]. Alive cells contain electrical signals for cellular communication, messaging, and regulation of nearly all biological system. In practice this can be modelled as a capacitor with action potential. Dead cells lack cytoplasm and has no charge carriers, so dead cells (and fat) will conduct the signal poorly. On the other hand, the salt content in the sweat result in improvement of the conduction of signals. Some of these effects can be minimized (degreasing, exfoliating, impedance matching), but due to the long timeframe of the WCE, it might be more advantageous compensating for these.

Link budget is a calculation of how much effect is received when parameters effecting the transmission (transmitted effect, noise, gains, losses etc.) is considered. As mentioned, if these parameters can't be found, approximations are done based on worst case scenarios and/or experiments. This field has been researched quite a lot when the signal goes through the body, but quite little when the radio signal is transmitted from inside the human body [94]. A research team in Malaysia has done this in the WCE case, and the results are summarized in Table 3.1 below. As it can be seen, the variation is between -9dB and -38dB in path loss (PL), depending on the different locations of the WCE. This can be especially effective if localization of the capsule can be done effectively, which is possible with quite high accuracy under ideal conditions [95].

Location	Region	Skin depth [cm]	Min Path loss	Max. Path loss
A	Upper Esophagus	7	-29 dB	
B	Lower Esophagus	13	-38 dB	
C1, C2, C3	Stomach	4, 10, 17	-10 dB	-34 dB
D1, D2, D3	Upper Intestine	5, 10, 15	-16 dB	-33 dB
E1, E2	Intestine (above abdominal region)	6, 14	-10 dB	-23 dB
F1, F2	Intestine (abdominal region)	5, 10	-9 dB	-14 dB

Table 3.1 Min and Max path loss in different regions at 450 MHz [94].

3.3 Receiver and Decoder

The receiver system consists of two main parts, a belt with antennas and a small computer to decode the signal. On the belt a matrix of 8 antennas, with sensor size of 40mm each, are connected by coaxial cables [83, 96]. Placement of the antennas help localize the position of the WCE, at any given time (illustrated in Figure 3.4 below). This placement structure also ensures that strongest possible signal is received at any given time and the different path losses in the different regions can be compensated (Section 3.2.2). The whole receiver system weights about 500grams, which includes the battery power supply. Patients can therefore move around freely, and they don't have to be connected to an external power supply, though they need to be cautious for extensive noise sources.

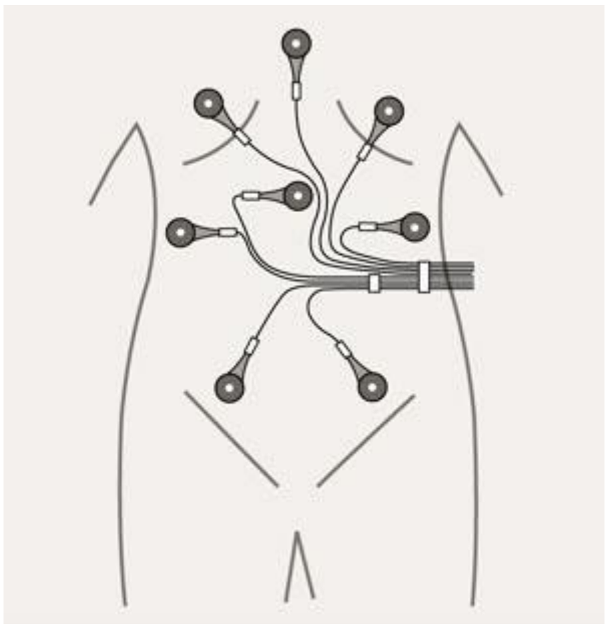

Figure 3.4 Placement of receiver antennas on the patient [96].

The actual decoder algorithm is very similar to the encoder, but as "unlimited" power in comparison, which means there is no computation limits. The transmitted signal is captured by the antennas and channel decoded (this part is omitted from the simulations), before the image decoding algorithm starts with the received bit stream. Received bit stream is decoded in the SR decoder (described in Section 2.6.3.1). Since the encoder has an embedded decoder in the DPCM structure (Chapter 2.4.3.1), the decoding process is the same as in the encoder. The exception is the quantizer which is done by inverse quantization (Q^{-1}) by (37). This results in the reconstructed error signal, $\tilde{e}(n)$, which is used to find $y(n)$ together with the previous reconstructed value, given by (16). If no bit error was introduced in the transmission and there was no quantization the decoded signal would be near identical to the encoded signal, $y(n) = x(n)$. Naturally, this isn't the case, so the value will be an approximated signal.

Since the closed-loop DPCM is dependent on previous values in order to obtain the current values, bit errors introduced earlier will propagate through the system. The RLE (SR) coding scheme has a variable code length, which is susceptible to bit errors, which will propagate to the DPCM. In other words, bit errors will appear and will affect the received image quality.

There are many ways of minimizing bit errors in a communication system. A common method is to include parity bits, to ensure that received bits are correct, and if they aren't, some can be recovered depending on how many are wrong (assuming parity bit is correctly transmitted) [97]. A specific way of using parity bits, is to use hamming codes, which is one of the first error-correcting codes[24] [98]. The general idea is to use parity bits calculated from the message, transmit as side information, and check if (and where) the error is located by recalculating (or use lookup tables) at the receiver. However, this (and most error correcting methods) require more energy and/or storage space at the encoder, as well as produce higher bit rate, which would be a disadvantage in the WCE case.

With WCE it would be very advantageous if error correction could be done at the receiver, without affecting the limited resources at the encoder. Two algorithms were proposed for this purpose [99]. The first method is based on simple error detection and concealment, which detects single pixel errors based on line detection. If bit error is detected, that particular pixel is discarded and reconstructed based on neighbouring pixels. This method is used if the number of bit errors is low. If the amount of bit errors is high, the second method is used, which discard the whole frame and reconstruct it based on inter-frame interpolation. This can be done since there is a temporal correlation between successive frames in a video sequence, and the discarded frame can be estimated with motion vectors. None of these methods affect the encoder performance, but will increase the SNR and bit-error ratio (BER).

3.4 Previous Work

As mentioned in the Introduction, this thesis is based on a project completed during fall 2015 [13]. This project explored the possibility of changing the image quality based on a location, which would be achieved by sending a simple signal back to the capsule from the receiver. Specifically, was the sampling rate in the MR part changed to not decimate and filtering the input signal in some parts, and increase it at other parts. This means that more of the high frequencies components were kept in the frames in the time they were coded with relative high quality. The result was an increase in PSNR by 4.7dB in high quality regions, and a decrease in PSNR by 9.6dB in low quality regions. The simulations showed that if the total time in high quality region was about 10-15% (approximately 1 hour out of a total of 8 hour) of the total time, this would decrease the total energy consumption. This could be very advantageous if the health care personnel can estimate the location based on other diagnostic procedures or symptoms.

The main goal was to investigate the possibility of adapting the parameters continuous and how much this would affect the required energy consumption. The results showed that this was possible and could be achieved by different triggers. Most obvious was a time, or a localization,

[24] In general, can normal parity code check for errors, but cannot correct, hamming code can also correct.

trigger, which could activate high quality frames depending on where the capsule was located (mentioned in Section 3.2.2) [95]. Other kind of triggers could for instance be analysis of the frames at the decoder, real-time control by physicians or a combination of different triggers. Other parameters could also be changed, for instance the predication coefficients, quantization steps or the framerate.

As already mentioned, previous work includes the initial design of the algorithm [7], post-processing error correction [99] and localization and tracking of the capsule [95]. These are described more in Section 3.1.1, 3.2.2 and 3.3, respectively.

4 Proposed Method

In this chapter a system overview will be first be given in Section 4.1, before an analysis of possible improvements of the current algorithm is presented in Section 4.2. Specifically, how to efficiently utilize a feedback loop from the decoder, back to the encoder. Following, will the proposed method for low complexity ROI coding be presented in Section 4.3. At the end will the limitations, implementation process and the evaluation methods be described in Section 4.4, 4.5 and 4.6 respectively.

4.1 System Overview

The system used for solving the problem of this thesis is based on the original algorithm, presented in Chapter 3.1. Originally, the algorithm was embedded, in the way that the encoded signal wasn’t decoded, but the required coefficients for analysis was extracted during the encoding process. This allowed for fast and efficient simulations for examining the most important part, the encoder. However, if the algorithm is to be implemented on an actual WCE, separation of the encoder and decoder would be a requirement. Separation have the advantage of being more straightforward, but simulations (in this case) will take longer time.

For simplicity, all of the required parameters are kept in a single “struct”. At the decoder, a new module is introduced which will update some of these parameters, as illustrated in Figure 4.1 below. Note that transmitting all parameters back to the encoder at each iteration, is not necessary and should be omitted in an actual implementation.

In the simulation process, different videos will be used to get a good understanding of how the image coder will work. Each video is read frame-by-frame, and each frame is encoded and decoded separately at each iteration of the simulation. At the entry of encoder, each frame is divided into the three RGB channels before the actual image encoder process starts. Note that since the simulation videos are compressed in a specific format, the results are expected to differ from a physical implementation.

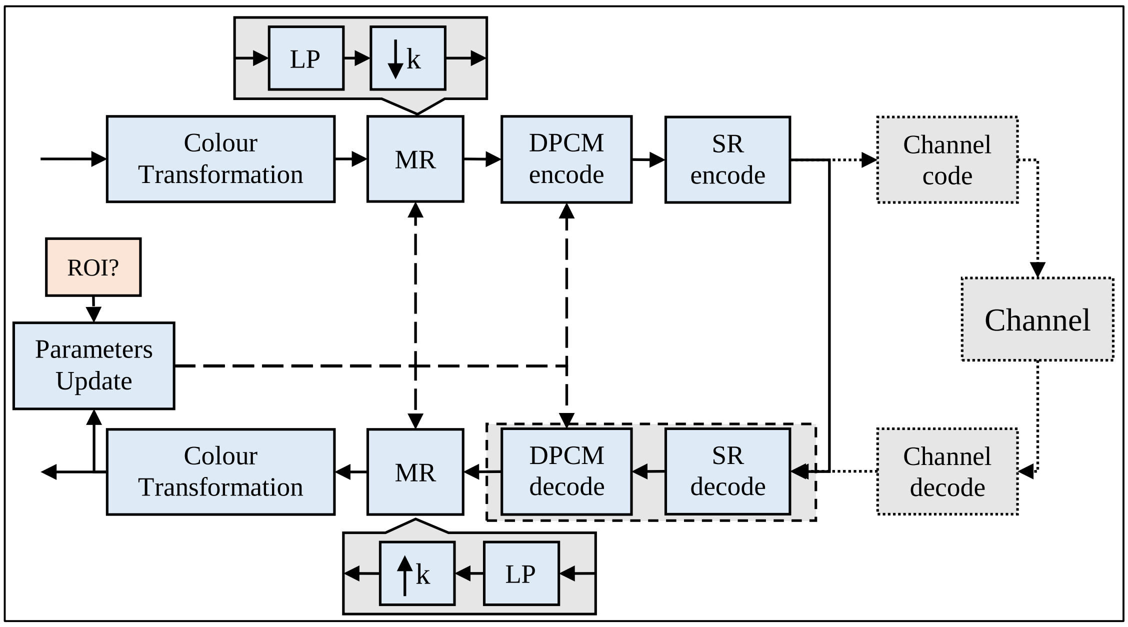

Figure 4.1: Overview over the modified algorithm

4.2 Feedback and Algorithm Analysis

Since the key aspect is to keep the algorithm minimized in required computational resources, many encoding techniques fail to become an option in this application. The YEF colour space (Section 2.3.1.3) on the other hand, has showed promising results in wireless endoscopic compression [50]. As the simplified YUV, it uses a base of 2 in the transformation matrix (3), which is beneficial in low power applications. Therefore, this will be implemented and compared to the existing YUV colour space.

Sending a return signal (feedback signal) with parameter updates is considered to be a cheap operation, since it won't require changes in hardware. There are different parameters which could be useful to update during the capsule's journey through the GI tract. As mentioned in Chapter 2.4.3.2, adaptive prediction filter could be of use.

Because the encoding algorithm should be kept as simple as possible, and the prediction coefficients are a simple averaging (38)-(39), only one value is chosen to be adapted. The relationship between the three prediction coefficients are kept unchanged. The chosen solution is to find the autocorrelation function (ACF) between y_t and y_{t+k} where $k = 0, \ldots, K$ is the lags:

$$a_k = \frac{c_k}{c_0} \tag{40}$$

Where c_0 is the sample variance, and c_k is given by:

$$a_k = \frac{1}{N-1}\sum_{n=1}^{N-k}(y_n - \bar{y})\,(y_{n+k} - \bar{y}) \tag{41}$$

In an image, the ACF is first calculated over all rows, giving a matrix where each row is an ACF of that row. Since there are three components to choose from, the luminance component is chosen as this contains the most information (Section 2.3.1). The average value over each column is thereafter calculated by (42), resulting in an ACF vector were each point is the average sum over that column.

$$\overline{a_k}(i) = \frac{\sum_{n=1}^{N} a_k(n,i)}{N} \tag{42}$$

By definition $a_0 = 1$, and in the AR (2) model, the prediction coefficient that is desirable is a_1. This would result in only one value needing to be transmitted back the encoder, a, in (39).

It could be argued that the quantization steps and/or dynamic range can also be made adaptive (Section 2.5.3). However, since all of the calculation has to be done on the output signal at the decoder (subject to transmission error, and lossy encoding), this isn’t expected to have a great impact on the resulting quality. DZ-quantizer uses the optimum uniform quantizer outside the DZ region, and force more values towards zero which is advantageous at very low bit rates. Since endoscopic images has very little variation between frames, BA quantizer would not increase quality a lot, while FA quantizer could. Unfortunately, FA quantization would increase computation (and energy consumption) too much at the encoder. Because of this, adaptive quantizer is omitted, and the focus will instead be on attempting to find a very low complexity ROI coding.

4.3 Very Low Complexity Region-of-Interest Coding

Initially, two schemes were proposed for ROI coding. The first was extracting the region, store it in a buffer, and code the frame as normal, before the ROI region was coded separately with higher quality (omitting the MR part). This would require extra storage/buffer, but it would remove the need for excessive parts of the image to be coded in higher quality. However, since the algorithm uses DPCM row wise, it would have fewer values to predict from. Early stages simulations showed as expected that the result would be of poorer quality in this ROI because of this. Therefore, this was discarded in favour of the other proposed scheme, which was built on the principle of adapting the values.

Previously, changes in the sample rate (and by that filtering) has been explored (Section 3.4), which showed promising results. Here the goal was to adaptively change the sampling rate in achieving higher image quality (the whole frame). Based on adaptively changing the sampling rate, this will be taken a step further by changing the sample rate within the frames in order to obtain a simple ROI encoding.

When the sample rate is set to be equal to 1, there is no decimation/interpolation or filtering, i.e. there is no Multirate part in the coder, and the colour transformed image goes directly into the DPCM coder. By adapting the rate in the middle of the image, some parts go through MR and some skip it, resulting in a grid structure showed in Figure 4.2. This ensures that the variables don't get filtered and decimated (a lossy process[25]), but instead keeps more of the high frequency components in that area. As stated in Section 2.4.2, most of the information in an image is in the low-pass part, but by omitting the high-pass parts the image quality will be reduced.

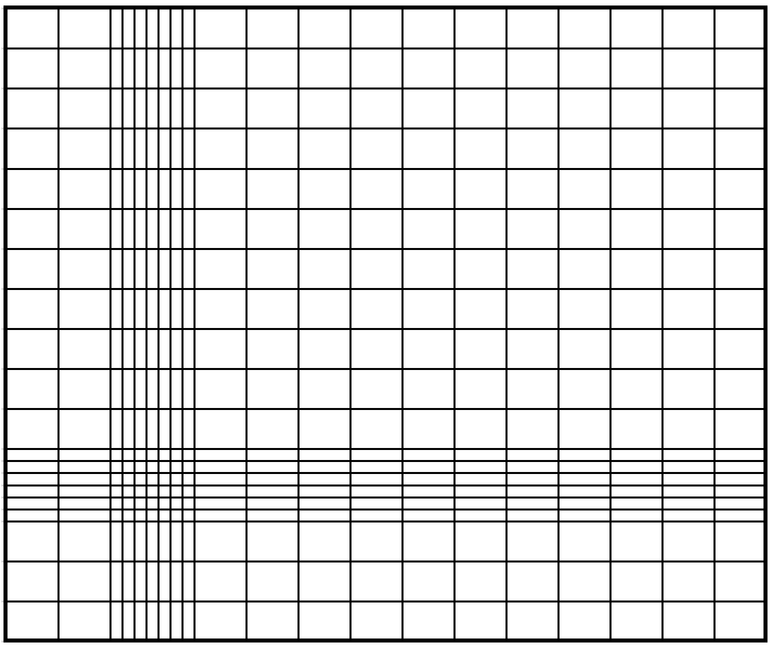

Figure 4.2 Structure of implantation of ROI area, after the MR part.

This method can be viewed as a simplified locally adaptive resolution (LAR) coding, but this LAR method is a little different. The LAR method is in general, variation in pixel sizes in the image depending on the activity in the image. It extracts the ROI area, and code it separately with a higher resolution (or don't code it), before the region is combined with the low-resolution image. However, while this technique is utilized to have a low rate transmission, the encoder is still quite complex in comparison of the suggested method [100-103].

The proposed ROI encoding scheme is expected to require more average bitrate per pixel, since more areas than just the ROI region will indeed be coded with higher quality (more high

[25] The total image coding process is still lossy due to the fact that the image is still quantized within the DPCM part.

frequency components). On the other hand, the encoding will be very simple (it doesn't require additional hardware implementation/space), and it doesn't require buffers or temporary storages.

To the authors knowledge, this ROI encoding hasn't been done exactly like this before.

4.4 Limitations

As already mentioned in Section 4.2, the feedback loop will, in the analysis part, be limited to the prediction filter coefficients. The simulations will be performed in MATLAB, which will have some drawbacks. MATLAB has no functionality for evaluation of the needed computational operations, and as a result it isn't possible to actually measure the required energy consumption. This will instead be estimated based on theory.

In ROI coding, only the implementation of the ROI region at the encoder will be considered, not finding the location of the particular region. The region can be found by image analysis (post-processing) by, for instance, looking at the light reflecting properties caused by variations in the haemoglobin protein (Section 2.1.3) [104]. This will be done at the decoder, resulting in a position vector to be transmitted back to the encoder. Some delay will therefore occur before the region is of higher quality, but it doesn't require delays or computations in the encoder.

4.5 Implementation

The chosen MATLAB programming language has some advantages since it is very simple, it has good matrix handling and it is reliable. Reliability is important for the ability to recreate every simulation result. On the other hand, MATLAB has some drawbacks as it cannot calculate the number of operations (and thereby complexity and energy consumption comparison), it isn't as memory efficient as C or C++, and is has some limitations in built-in functions. An example of limitations in MATLAB functions is the built-in function VideoWriter used in the simulations, which cannot handle adaptive framerate. It is possible to use a custom-VideoWriter class called "QTwriter", but this is used for QuickTime video format (.mov), and the results won't be directly comparable with the audio video interleaved (.avi) format [105]. However, changes in the framerate is directly related to how often the camera sensor captures the frames, and the encoding process will be the same. A feedback loop will allow adaption of the framerate, so the direct implementation of this is omitted.

An important feature in the implementation, is the ability to recreate every simulation result as well as separation of the different blocks (Figure 4.3). Separation of the blocks will result in some repetitive tasks, since the performance evaluation will in fact have to partly code both the original frame and the reconstructed frame. This will make the total simulation process slower, but it won't affect the encoding/decoding performance.

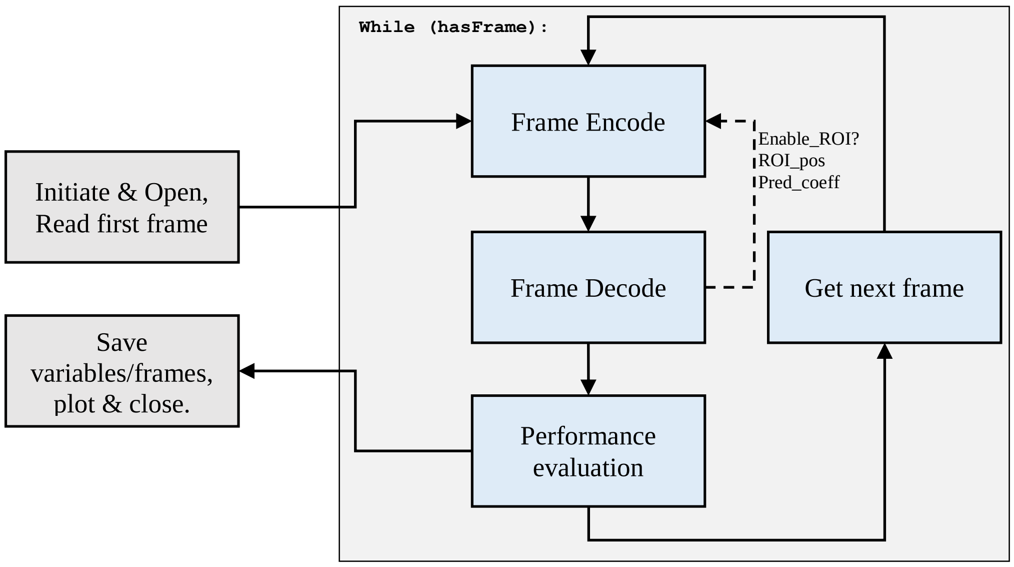

Figure 4.3 Overview of implementation of the simulation system

As mentioned in the previous chapter, the ROI will be coded by a position vector transmitted to the encoder. The vector is defined as:

$$ROI_{pos} = [x, y, w, h] \tag{43}$$

Where *(x, y)* is the pixel location, and *w*, *h* is the width and height of the area respectively. The region is rectangle shaped, with top left corner as the starting point *(x, y)*. This means that a total of only five integer values (including the prediction coefficient) has to be transmitted, as well as one logical value (active ROI encoding, true/false).

4.6 Evaluation

In image encoding it is common to distinguish between objective and subjective measurements of the image quality, since these aren't necessarily the same. Similarly, both subjective and objective evaluation will be performed in the thesis, but the subjective quality measurement will be evaluated by the author, together with sample images in the thesis for the reader to be able to evaluate as well. Subjective quality measurement will follow the principles of mean opinion score (MOS)[26], with reference to the source (input) image. The range is from 1 to 5, and is defined as bad (1), poor (2), fair (3), good (4) and excellent (5) [37].

[26] In this case, not many people will be asked, so will only consist of authors opinion score (OS).

Two main quality evaluations will be done for the objective analysis; Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). PSNR will be performed for the objective distortion for each of the three colour channels, before summed to the total CPSNR[27] (colour peak signal-to-noise ratio).

$$PSNR_{(\cdot)} = 10 \cdot \log_{10}\left(\frac{255^2}{\frac{1}{I \cdot J}\sum_{i=1}^{I}\sum_{j=1}^{J}\left(Y(i,j) - \hat{Y}(i,j)\right)^2}\right) \tag{44}$$

Where (·) indicate the colour component, and *(i, j)* is the pixel location. SSIM is an image quality metric in which the three characteristics of the images is considered; Luminance, contrast and structure [106]. The measurement is performed between the distorted (coded) image, *x*, and a reference picture, *y*, given by (45). SSIM gives a measure ranging from 0 to 1, where *SSIM=1* if the two images are identical, and *SSIM=0* if no parts of the image is similar.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{45}$$

Where μ_x and μ_y are the local means, σ_x and σ_y is the standard deviations and σ_{xy} is the cross covariance for the images *x*, *y*. C_1 and C_2 are constants to avoid instability when the local means or standard deviation is very close to zero, and are related to the dynamic range[28] L of the pixel value (46). L is typically 255 for an 8bit greyscale image, and is multiplied with a small number $K_1 \ll 1$.

$$C_1 = (K_1 \cdot L)^2 \tag{46}$$

Two evaluation methods will be performed to get an idea of the energy consumption of the image coder, the average bit-per-pixel (bpp) and the compression ratio (CR) given by (47)-(48). Both of these are common in image encoding evaluation and will give an indication of the energy requirements of the algorithm.

$$R[bpp] = R_{ch1} + R_{ch2} + R_{ch3} \tag{47}$$

$$CR = \left(1 - \frac{R_{c1} + R_{c2} + R_{c3}}{8+8+8}\right) \cdot 100 \tag{48}$$

Where $R_{(\cdot)}$ are the average rate for each of the colour components, assuming 8bit per colour channel on the input image.

[27] Also known as PSNR $_{\text{overall}}$

[28] Not to be confused with the *dynamic range* of a quantizer.

5 Results

In this chapter the results from the simulation will be presented, in order for a conclusion to be formulated. The results are divided into two parts; Section 5.1 covers the algorithm and feedback analysis, while Section 5.2 covers the ROI encoding.

Both parts are the result of simulations with different videos. Specifications about the simulation videos can be found in appendix A.2.

5.1 Part 1: Algorithm

In part 1, the algorithm is analysed using different settings and methods. Each is compared with the original algorithm in this section.

By analysing the algorithm, the areas with the most potential was considered to be changing the colour space and adapting the prediction coefficient. These areas don't require additional buffer or storage, since every calculation is performed at the decoder, i.e. the complexity is unchanged. This is the BA configuration is expected to give poorer result than FA would, but it reduces the power consumption which is the main limiting entity in the device.

Note that even though this was considered the area with the most potential, it wasn't the only area changed in the algorithm. As mentioned in Chapter 4, the algorithm was separated, and some functions had to be rewritten to allow encoding of ROI. For instance, the MR part had to be changed, to not include any built-in functions, and this needed to be done manually.

The evaluation of the modified algorithm is performed as described in Section 4.6. Since many results are very similar in structure, the graphs for each one is considered redundant, but the most important ones will be included (this doesn't include the results which will be summarized in tables). The difference between them is usually small variations in the axis.

5.1.1 Simulation 1

The first simulation was performed with a short (~2s), quite low resolution (241x401) and low quality video sequence (Appendix A.2.1). This video was chosen since it achieves quick simulations due to the few number of frames, and would be similar in resolution to the current WCE.

The algorithm was executed with a constant sampling rate in the MR part, but different between the colour components. Since the luminance has the most information, this has a sampling rate of 2, while the chrominance components are decimated/interpolated by 3. Figure 5.1 below show the performance of the original algorithm, which uses the YUV colour space and a fixed prediction coefficient, *a=0.8*.

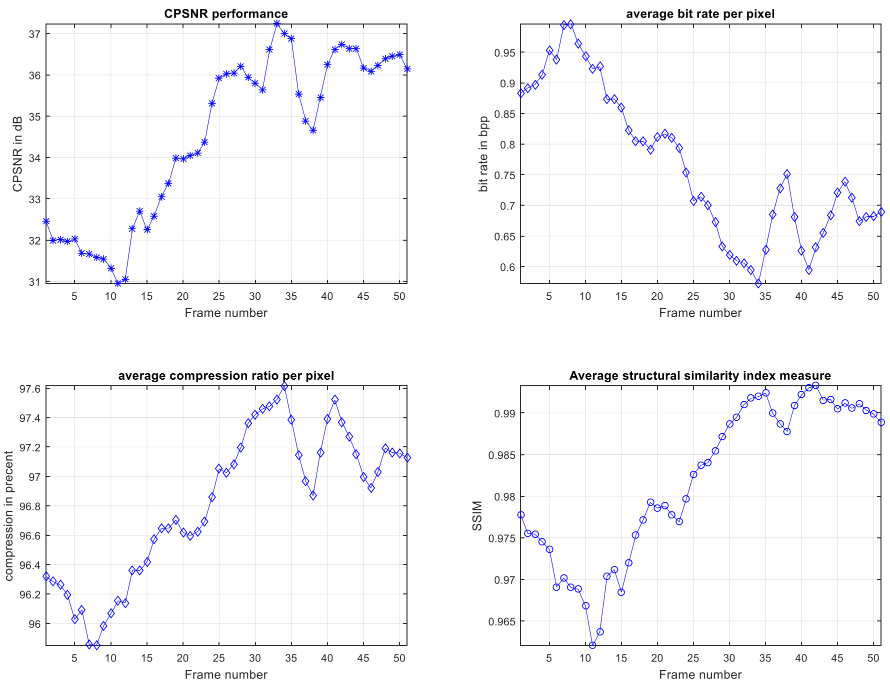

Figure 5.1 Performance of original algorithm

As can be seen from the graphs above, the performance increases towards the end, which is expected in predictive coding schemes. This particular simulation video has less variation in the colour components towards the end than in the beginning (more white light reflection). Notice a sudden drop in all of the measurements around frame number 35, which is most likely due to a change in camera angle resulting in changes in reflected light intensity.

In the modified algorithm, different parameters was tested, but the one shown in Figure 5.2 below was with the YEF colour space, and adaptive prediction coefficient. Observable in the figure, is the drop around frame number 35 gone in the CPSNR and SSIM measurements, but the same in bitrate per pixel and CR. However, even though the drop is approximate the same (~0.8% in CR in both algorithms), the overall performance is increased, as the value drops close to the original algorithms' value before the drop.

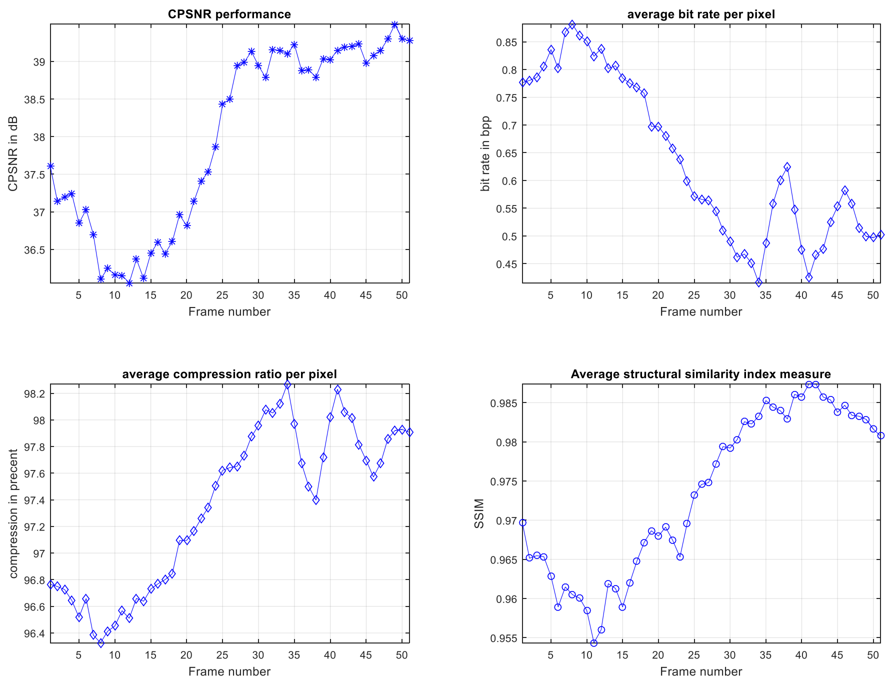

Figure 5.2 Performance of modified algorithm, with YEF & adaptive prediction

The drop in CPSNR and SSIM, is due to the chosen colour transform. Simulations which uses the YUV colour space has this drop, but not the ones using the YEF colour space. This could indicate that the YEF is more suited for endoscopic images, but it could also be just this video sequence that behaves like this. Adaptive prediction seems to mainly affect the values of the axis, not the graphs themselves. Figure 5.3 shows the variation of the prediction coefficient over the sequence.

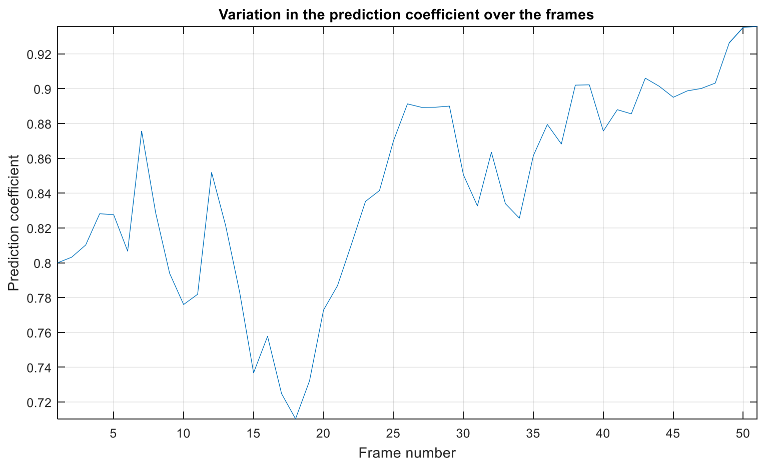

Figure 5.3 Variation in the prediction coefficient over the frames

The results from the different simulation settings is summarized in Table 5.1 below. As shown in the table, the CPSNR has increased quite a bit from the original algorithm. The YEF colour space generally seems to require less average bits per pixel than the YUV colour space and by that better compression ratio, but the SSIM has slightly decreased. The reduction in SSIM might be the result of the YEF colour space removes more of the red components in the image compared to the YUV, as shown in Figure 5.4 below.

Settings	**BPP [bpp]**	**CR [%]**	**CPSNR [dB]**	**SSIM**	**OS**
Original algorithm	0.76527	96.8114	34.49	0.9818	3.3
YUV, a=0.9	0.71062	97.0391	37.4652	0.98244	3.2
YEF, a=0.9	0.62778	97.3843	37.8722	0.97367	3.3
YUV, a=0.8	0.76527	96.8114	37.425	0.98246	3.5
YEF, a=0.8	0.64754	97.3019	37.8484	0.97304	3.6
YUV, a=adaptive	0.73026	96.9572	37.5166	0.98254	3.5
YEF, a=adaptive	0.63738	97.3442	37.982	0.97341	3.6

Table 5.1 Simulation results from video 1.

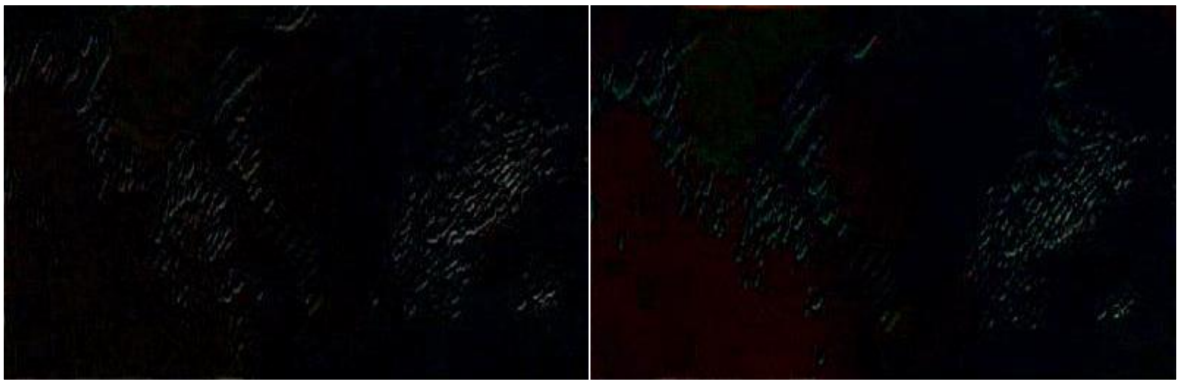

Figure 5.4 Difference between reconstructed and original frame. YUV colour space to the left, YEF to the right.

5.1.2 Simulation 2

The second simulation was performed with a video which had longer duration (~97sec) and higher resolution (576x768 pixels) (Appendix A.2.2). The video is taken from an endoscopy (not capsule) procedure, meaning a lot higher quality and more similar to RAW input image.

By the looks of it, the tissue is healthy, but a bit of fluid is seen, which reflects more light. A black boarder surrounds the image, and a few tags are displayed in the frames as well, see Figure 5.7 below. Note that the simulation video had a few empty frames at the end, so the last few frames was omitted from the simulation. The performance of the original algorithm is displayed below, in Figure 5.5.

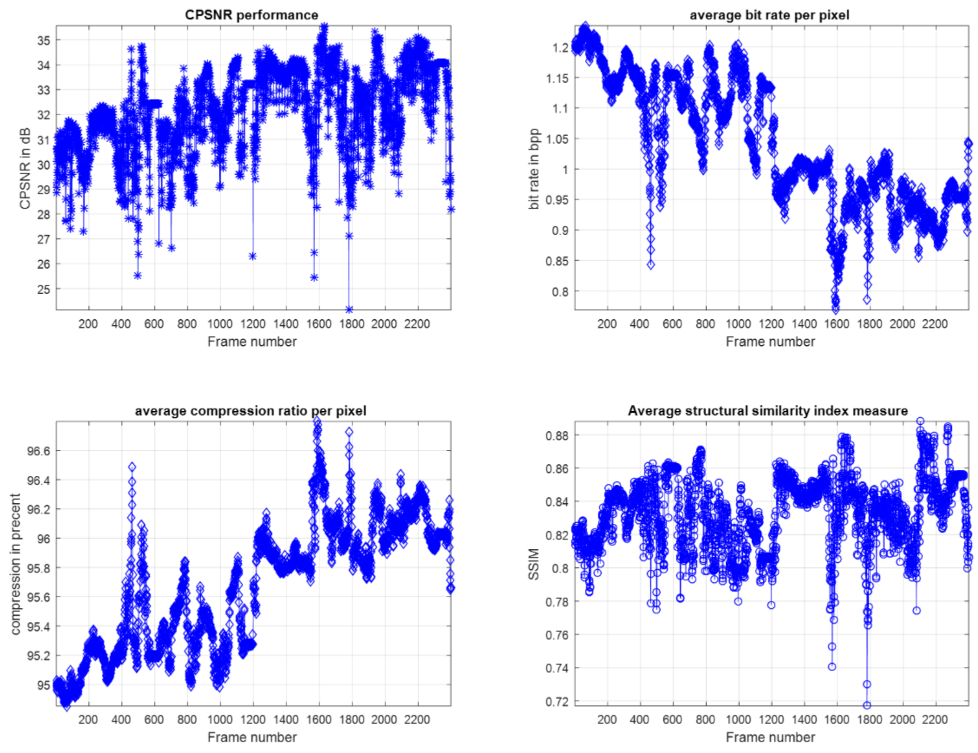

Figure 5.5 Performance of original algorithm with second simulation video

As with the previous simulation, the performance increases towards the end, but with a few spikes. This might be since the video freezes at some points resulting in static frames (increase in performance), before the whole frame is shrunk down to the corner (high variation, decrease in performance). This happens a few times when the operator of the endoscopy is capturing the frames.

The structure of the performance measurements from the modified algorithm, is very similar to the structure of the original algorithm, as seen in Figure 5.6 below. On a closer look, it’s clear that the graphs are scaled (increased), which also reflects the average results shown at the end of the subchapter in Table 5.2.

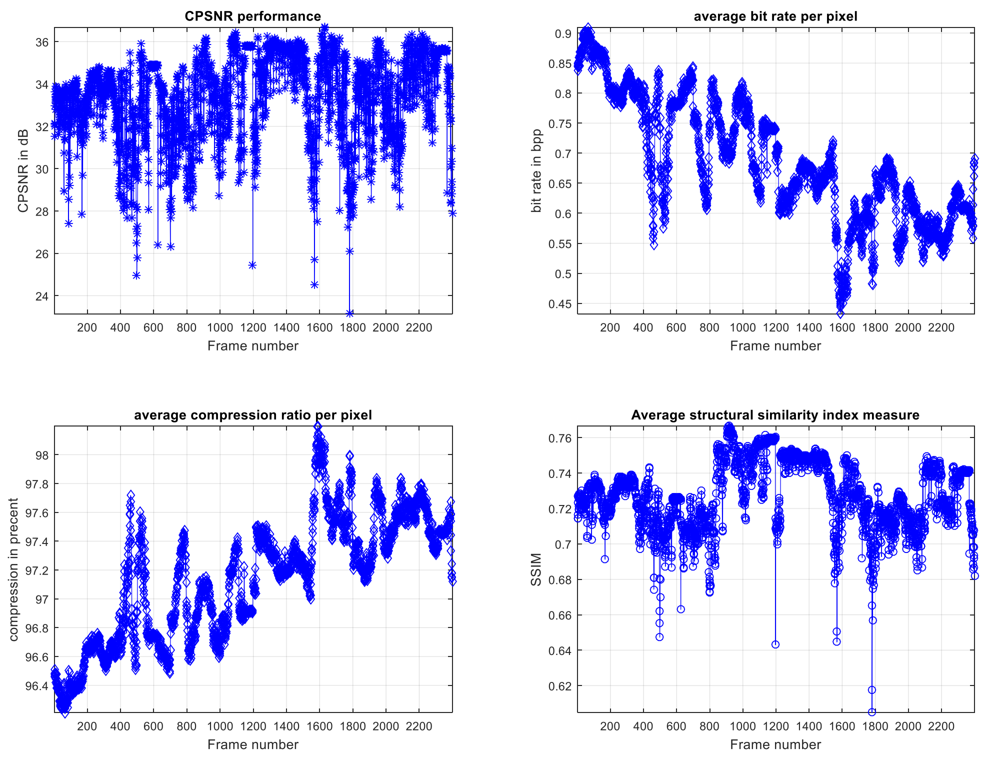

Figure 5.6 Performance of algorithm with YEF & adaptive prediction

Analysing the reconstructed movies, reveals that in the adaptive performance, a few bit errors occur from the bright spots in some of the frames. These are explored more in the next simulation (Section 5.1.3), where they are a lot more frequent. In this section, similarities and differences in the reconstructed frames will be analysed further, and the source frame can be seen below in Figure 5.7.

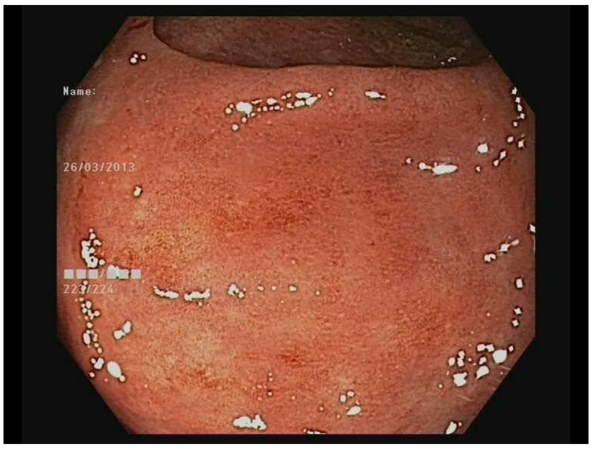

Figure 5.7 Source frame

Figure 5.8-5.11 below, shows the reconstructed frames with the different settings, as well as the resulting frame from the original algorithm. The first noticeable difference is in the black areas in the frames, where the YEF colour space produces red components mixed with black. The result is that true black areas aren't black. The black boarder around the frames, wouldn't be there in a WCE implementation, since the camera sensor doesn't have an obstruction in front of it. Most of the black areas would be from further down the GI tract where the reflected light would be too weak to be able to see before the capsule arrives to that section. Based on this, the colour transformation isn't considered to have any major limitations compared to the YUV space.

By a closer study, it is observable that the tissue areas in the middle, are a little greyer than in the YUV colour space. On the opposite side, on the top right corner (of the tissue) the area is greyer/greener in the frames with YUV space than in the YEF which are slightly red in that area.

It is also observable that some of the darker regions in the middle of the tissue is slightly larger with a prediction coefficient equal to 0.8 than with adaptive. Nonetheless, this is very difficult to see in the in the unscaled versions, where it is easier to see the same regions in all of simulations.

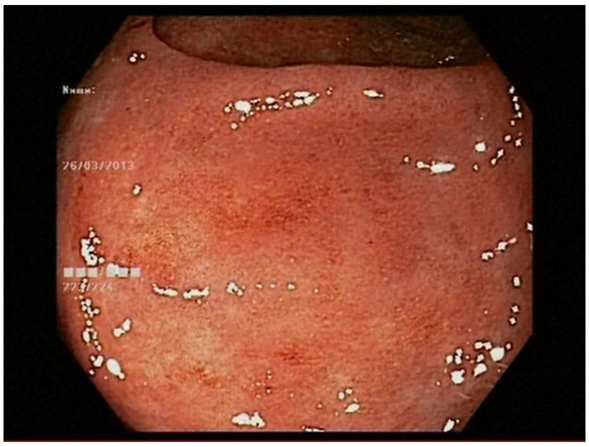

Figure 5.8 Reconstructed frame from original algorithm

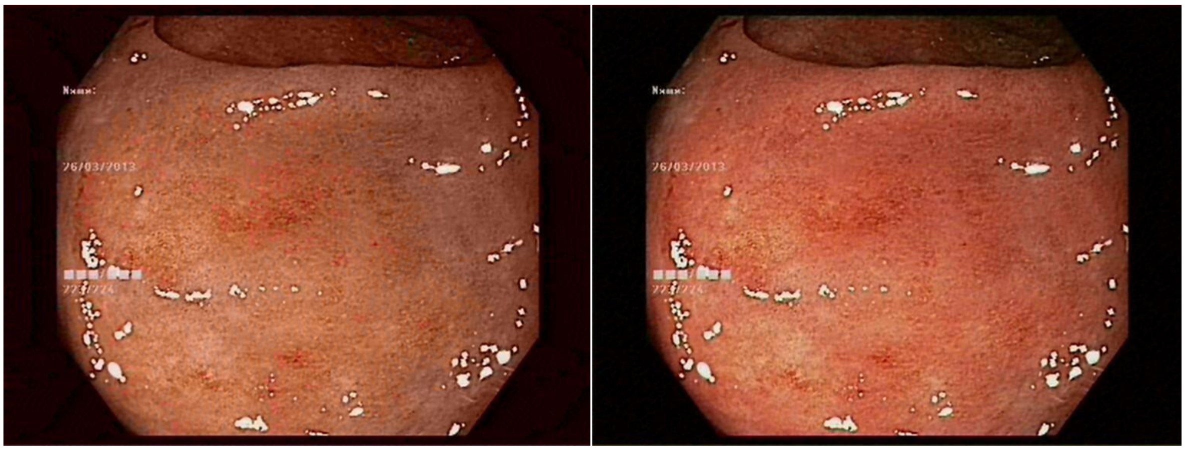

Figure 5.9 Reconstruced frames with *a*=0.8. YEF to the left and YUV to the right

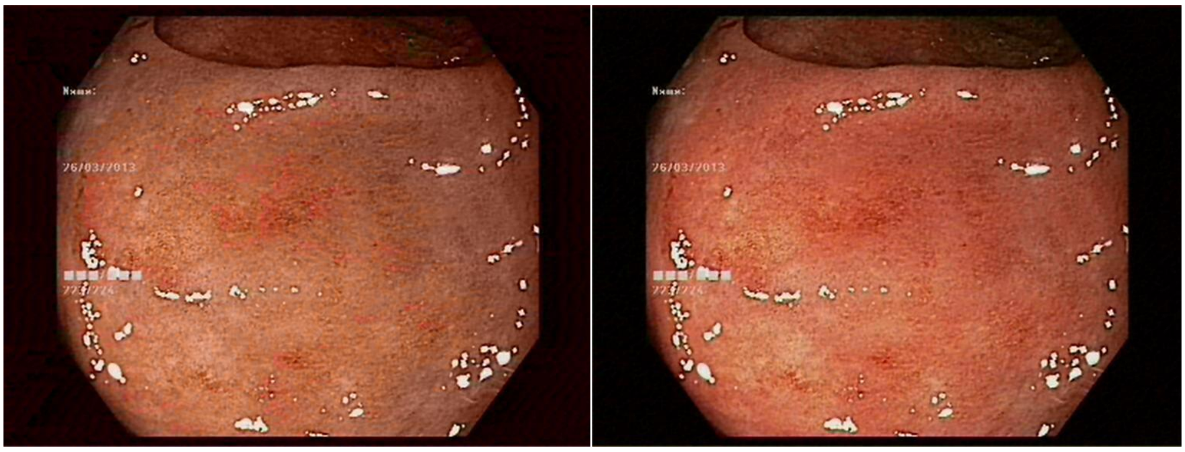

Figure 5.10 Reconstruced frames with *a*=0.9. YEF to the left and YUV to the right

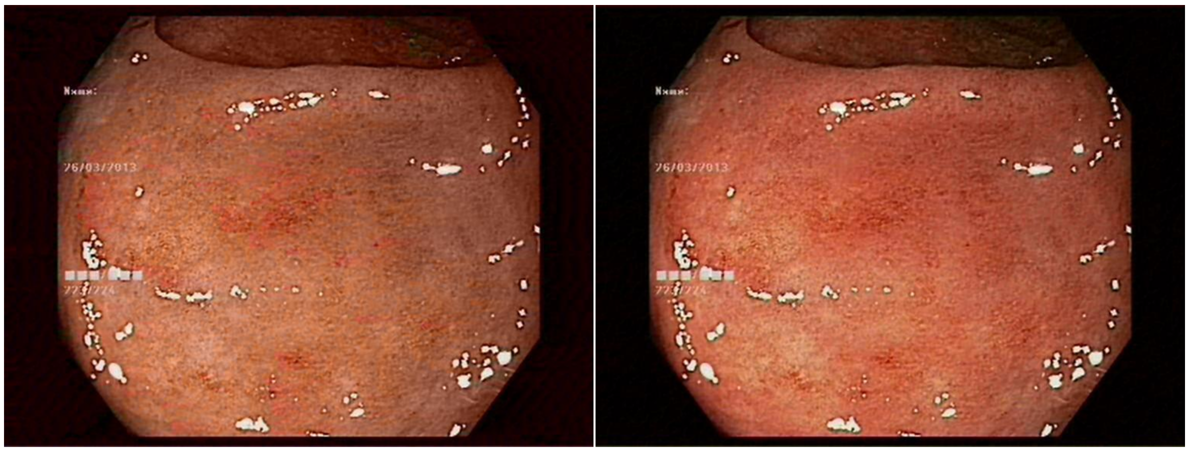

Figure 5.11 Reconstructed frames with adaptive prediction. YEF to the left and YUV to the right

It is up to reader to decide which will have the highest subjectively score, which can be compared to the subjective score from the author. Notice that it is easier to see differences in the reconstructed videos than on the scaled frames (~40% of original frame size).

The objective scores (along with the authors opinion) are summarized in Table 5.2 below. Most noticeable is the major difference in the average bitrate per pixel (and by that compression rate), where 0.35bpp separates the best and the worst performing settings. At the same time, this is considered among the highest scoring in the author's subjective score. Generally, the YUV has worse performance in energy consumption, but slightly better quality, than the YEF. Which isn't the subjective perceived opinion.

Settings	**BPP [bpp]**	**CR [%]**	**CPSNR [dB]**	**SSIM**	**OS**
Original algorithm	1.0414	95.6607	32.1601	0.83279	3,0
YUV, a=0.9	0.84076	96.4968	34.2188	0.85463	3,2
YEF, a=0.9	0.70575	97.0594	33.3165	0.72331	3,3
YUV, a=0.8	1.0414	95.6607	34.15	0.83154	3,3
YEF, a=0.8	0.83594	96.5169	33.2427	0.71674	3,5
YUV, a=adaptive	0.7895	96.7104	34.1763	0.85975	3,3
YEF, a=adaptive	0.68755	97.1352	33.3114	0.72674	3,5

Table 5.2 Simulation results from video 2

5.1.3 Simulation 3

The third simulation was with a cropped video of a normal (healthy) oesophagus. Originally, this video contained surrounding black area (same as the second simulation), and the goal was to see how this would affect the simulation results. The simulation video is between the first and second simulation video, in both length (~29.6 sec) and in resolution (430x378), details can be found in Appendix A.2.3. As the specifications show, this video sequence has the MPEG-4 format (.mp4) which is different from the previous videos. Because of this, the simulation video was also converted into the audio video interleave (.avi) format, in order to check if there were any deviations in the results between these two formats. Simulation with the two different formats was identical, therefore only one will be presented. The same variation in settings/parameters is applied here, as in the previous two simulations. Figure 5.12 below show the performance of the original algorithm.

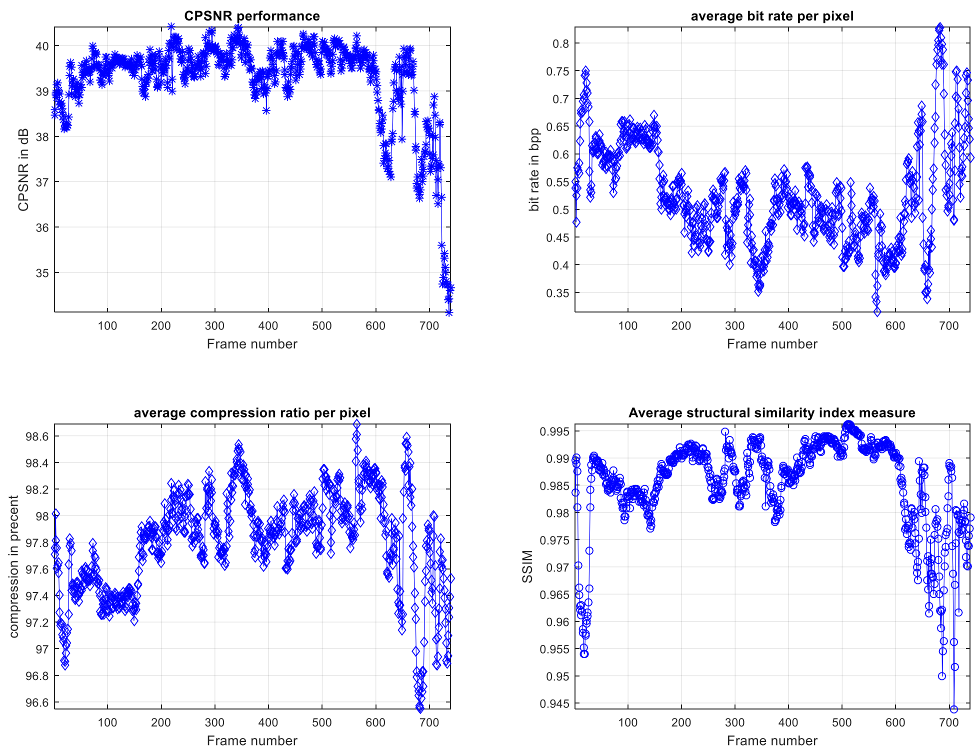

Figure 5.12 Simulation result from original algorithm

Notice the high variation, and the drop in performance towards the end. This is expected since there is high variation in the simulation video in this area. The endoscope passes from the oesophagus and into the stomach. High difference in regions tends to decrease the performance of predictive coding, since it relies on past samples to predict current one.

This particular simulation video has another feature, as it has quite a lot of different “errors”. It was still chosen since the capsule can be exposed to the same errors. Errors in the simulation video include bit errors, fluid hitting the camera, black frame, loss of focus and variation in light intensity. Interestingly, at frame number 500, the reconstructed frame in the YEF colour space is completely black (SSIM=0, Figure 5.13), but not in the YUV colour space. In the source, that frame is very blurry, and it is impossible to see anything else than a monotone grey image. This is also the case in YUV, but not in the YEF simulations (independent of the prediction coefficient).

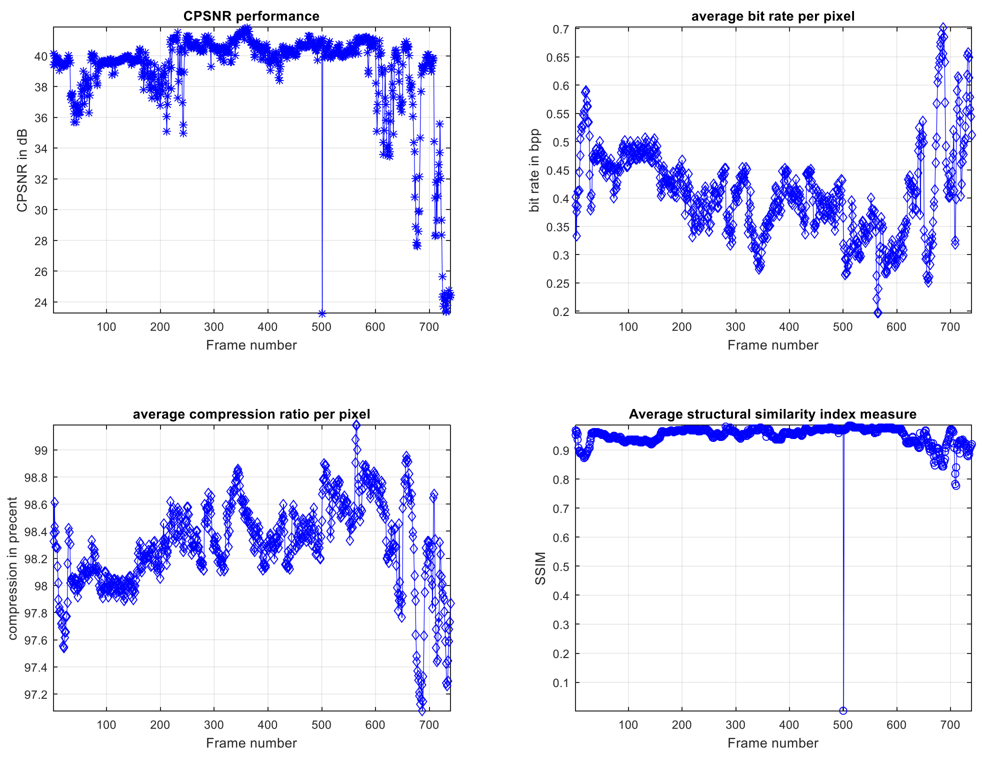

Figure 5.13 Simulation results with YEF & static *a*=0.8

All of the simulations with this video sequence, produced quite a bit of corrupted pixels, especially in the luminance channel. These corrupted pixels propagate to neighbouring pixels, and form square edges in horizontal and vertical lines, as shown in Figure 5.14. These can be reduced a lot when applying error concealment, described in Section 3.3 [99].

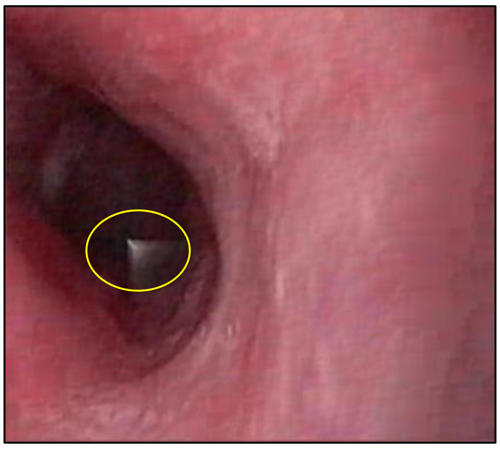

Figure 5.14 Corrupted luminance pixel

In adaptive prediction coefficients, the number of corrupted pixels, and the size of the errors increases quite a bit. Naturally, this happens since corruption propagates through the frames more, and the prediction coefficient is calculated from the luminance component (Section 4.2). This is very much reflected in the graphs of the performance of the adaptive prediction shown in Figure 5.15 below.

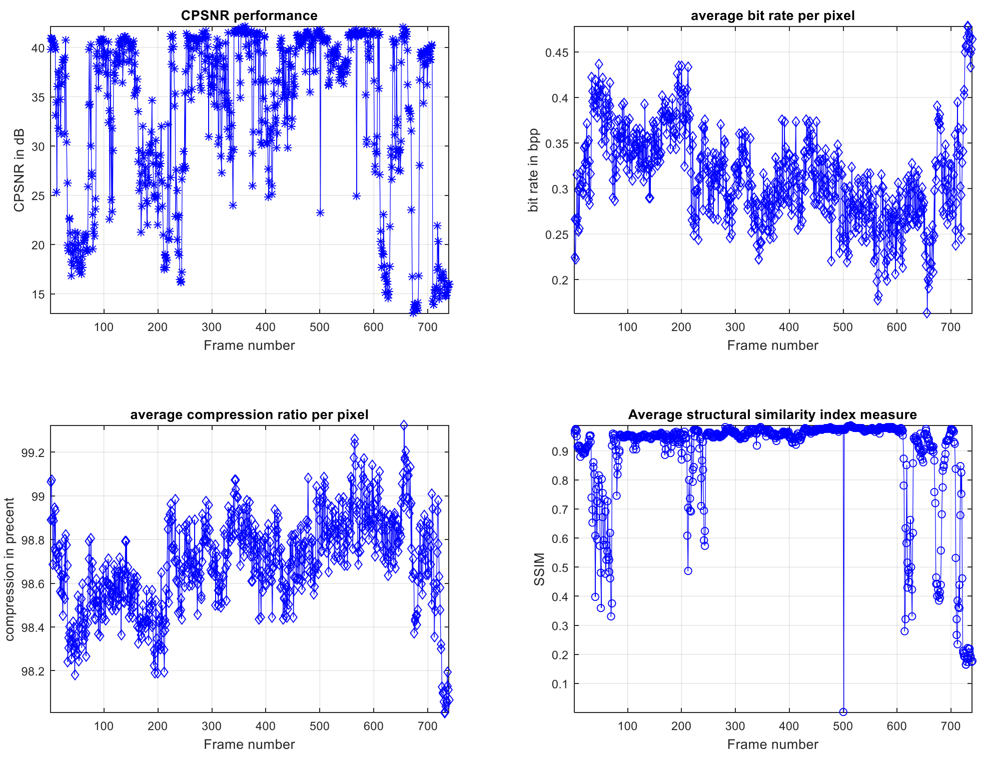

Figure 5.15 Simulation results with YEF & adaptive prediction

Table 5.3 below summarizes the different simulations done with video sequence 3. Note that in the simulations with adaptive prediction, the quality seems better as long as no pixel is corrupted (some frames). Unfortunately, this is few frames, making the overall score (both objective and subjective) lower than for the simulations with constant prediction coefficient.

Settings	**BPP [bpp]**	**CR [%]**	**CPSNR [dB]**	**SSIM**	**OS**
Original algorithm	0.52825	97.799	39.2525	0.98537	2,9
YUV, a=0.9	0.36903	98.4624	37.6099	0.97915	2,9
YEF, a=0.9	0.28866	98.7972	38.2481	0.94798	2,9
YUV, a=0.8	0.52825	97.799	38.9714	0.98367	3,2
YEF, a=0.8	0.40995	98.2919	39.0113	0.94866	3,3
YUV, a=adaptive	0.35975	98.501	31.9438	0.87474	2,3
YEF, a=adaptive	0.31426	98.6906	33.4076	0.8843	2,2

Table 5.3 Simulation results from video 3

5.1.4 Simulation 4

The fourth simulation was performed to investigate the impact the simulation videos would have on the objective scores. Since the simulation videos is also compressed, this could affect the objective results quite a lot. To examine this, the reconstructed video after simulation 1, was recompressed. Table 5.4 below shows the result of compressing the compressed video with different settings.

Settings	**BPP**	**CR**	**CPSNR**	**SSIM**
Original algorithm	0.64069	97.3304	36.4843	0.98888
YUV, a=0.9	0.6181	97.4246	39.7259	0.98905
YEF, a=0.9	0.5437	97.7346	41.231	0.98776
YUV, a=0.8	0.64227	97.3239	39.7516	0.9894
YEF, a=0.8	0.54038	97.7484	41.2313	0.9884
YUV, a=adaptive	0.61689	97.4296	39.8196	0.9893
YEF, a=adaptive	0.53949	97.7521	41.3225	0.98828

Table 5.4 Simulation results after compressing compressed video

Comparing Table 5.4 and Table 5.1, is it clear that the objective measures have increased in the rerun. CPSNR has increased between 2dB and 3.5dB, bitrate per pixel has decreased with approximately 0.11bpp and SSIM has increased by around 0.15. The subjective evaluation is difficult, as it is quite hard to distinguish between the compressed and recompressed images, as displayed below in Figure 5.16. Especially, when the whole video sequence is evaluated and not only single frames.

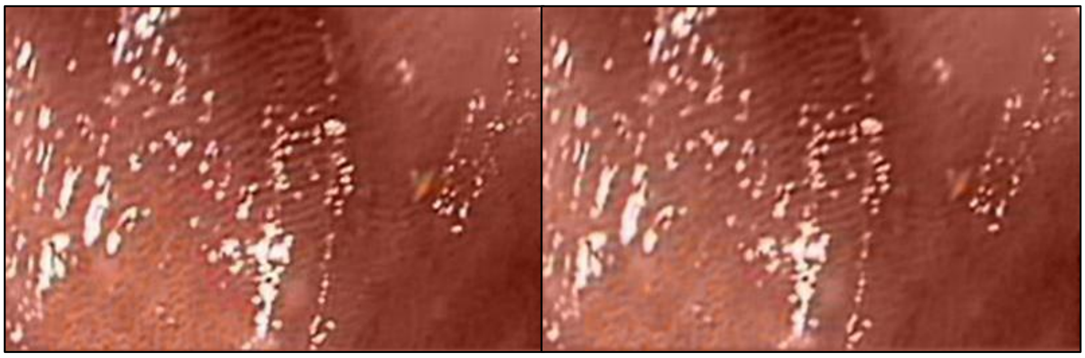

Figure 5.16 Comparison between compressed (left), and recompressed (right) video. Settings: static prediction at 0.8 and YEF colour space.

5.2 Part 2: Region-of-interest (ROI)

In this part, the results from implementing ROI encoding will be presented. Simulation of the implementation is performed with each of the three simulation videos, with the YEF colour space and adaptive prediction as the general settings. The results will afterwards be discussed in Chapter 6.2.

The MATLAB code can be found in Appendix B. Note that the code isn’t optimized, and it is possible to reduce the number of calculations performed. For this thesis, the important part was to test if the proposed method would work and how good the results would be.

When analysing the results from the encoding, the first to look at was the difference between the reconstructed frame and the input frame. Figure 5.17-5.19 shows this difference in the first video (Appendix A.2.1), with YEF and YUV respectively. The YUV was included for viewing purposes, since the YEF include red components in the difference. It is observable that inside the ROI area, no high frequency components are shown. On the horizontal and vertical axis of this region, less of the high frequency regions are shown than the surrounding areas of the frame. This was as expected for the low complexity ROI coding, see Figure 4.2 in Section 4.3.

Notice that these partly high quality coded axis was more observable in the moving video sequence than in still images.

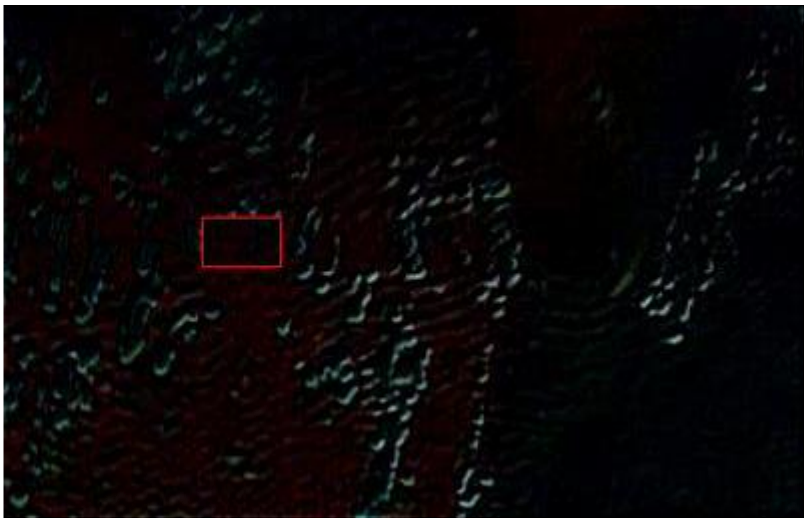
Figure 5.17 Difference between reconstructed and input frame video 1, with ROI coding, YEF & adaptive prediction.

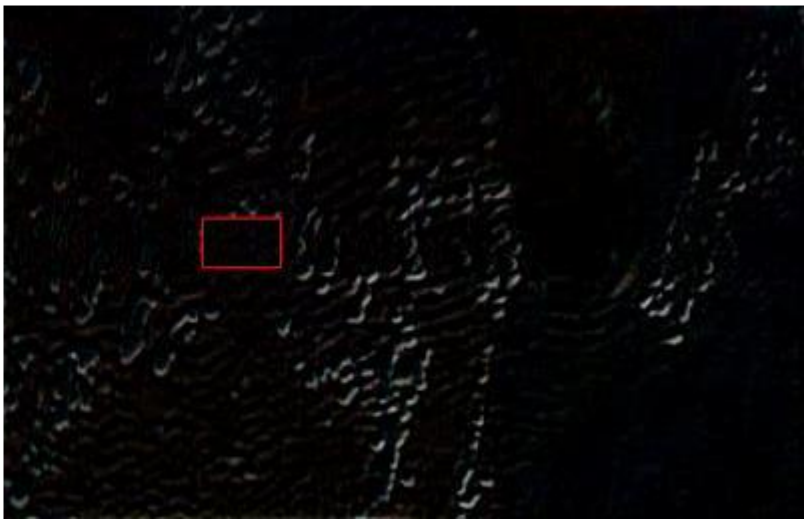
Figure 5.18 Difference between reconstructed and input frame video 1 , with ROI coding, YUV & adaptive prediction.

The ROI area is chosen according to (43), with values:

$$ROI_{pos} = [100\ 100\ 40\ 24] \qquad (49)$$

In other words, the top left corner of the region is located at *(100, 100)* with a width of 40 pixels and height of 24 pixels. This is 10% of the frame size for this video. If the frames are compared to Figure 5.4, which shows the difference without ROI coding, some deviation is observable. Outside the ROI area, it seems that more of the high frequency components are filtered than before. This also shows in the CPSNR (33.12dB) and SSIM (0.9553) which is actually lower with the ROI encoding.

Note that average bitrate and compression ratio won't be calculated here. Since the script calculates averages depending on the sampling rate over the whole frame, it would be a bit complicated to correctly compute this effectively the way it is implemented at the moment. On the other hand, it is possible to estimate the increase based on the increase in resolution. With that ROI sized area, additional samples kept in luminance component is 9600 samples, and 12800 samples in each of the chrominance components. This results in a total of 35200 additional values to be coded and transmitted per frame as shown in (50), corresponding to 12.2% increase.

$$\left(\frac{40}{2}\cdot 240+\frac{24}{2}\cdot 400\right)+2\cdot\left(\frac{2\cdot 40}{3}\cdot 240+\frac{2\cdot 24}{3}\cdot 400\right)=35200 \qquad (50)$$

The simulation results from the other two videos shows similar results as the first one; deviation from part 1. CPSNR and SSIM decreased in the second video, however, in the third video the CPSNR and SSIM increased quite a lot. The CPSNR increased by nearly 6.9dB and SSIM was raised by 0.072. When investigating the reconstructed movie, it showed that the amount of bit errors was drastically reduced compared to part 1, as seen comparing Table 5.3 and Table 5.5.

Simulation	**CPSNR**	**SSIM**
Video 1	33.1205	0.9553
Video 2	30.5174	0.71724
Video 3	40.2831	0.95677

Table 5.5 CPSNR & SSIM results from ROI encoding

If the subjective opinion is considered instead of the objective the results are generally better. It is mostly very difficult to observe the reduction in the videos, except for fewer bit errors. This can be seen when comparing two frames together, as shown in Figure 5.19, below. The text "Name:" is actually readable since it is in the ROI area, but it isn't readable in the frame from simulation 2 (Section 5.1.2). In some very fast moving sections of the videos, it seems that the proposed scheme has some problems, where the axis of the region show in reconstructed image as well (similar to the difference shown in Figure 5.17).

It is presumed from this, that there might be a small problem with the implemented performance evaluation and/or some other part of the implementation process. This topic will be discussed more in Chapter 6.2.

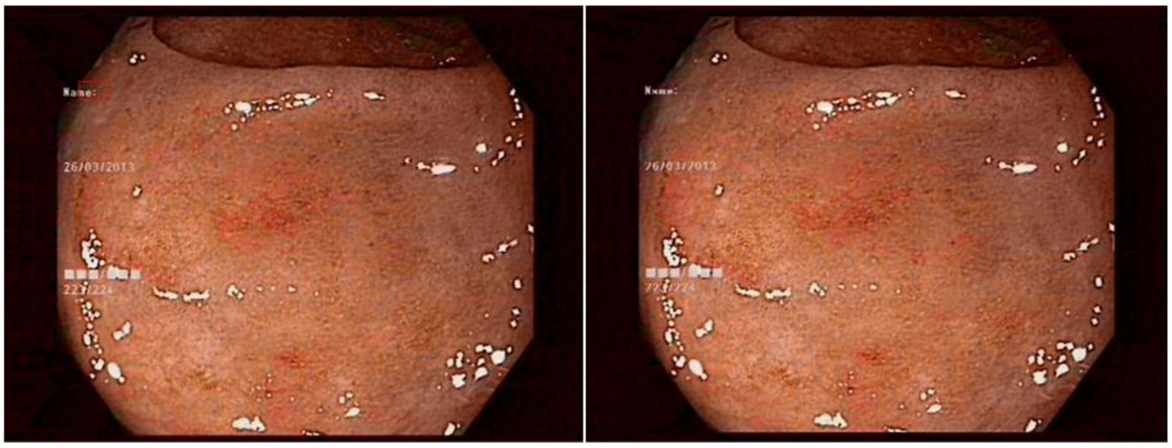

Figure 5.19 Comparison of reconstruced image with and without ROI. ROI encoding to the left, and without to the right. Both with YEF and adaptive prediction.

6 Discussion

In Sections 6.1-6.2, the results from Chapter 5 will be discussed, before a brief complexity and sources of errors analysis is presented in Sections 6.3-6.4.

The proposed method will thereafter briefly be compared with other WCE schemes in Section 6.5. In Subchapter 6.6, a brief comparison with video encoding instead of image encoding is discussed, before hardware implementation is discussed in the subsequent section (6.7).

6.1 Part 1: Algorithm

Analysing the results from Section 5.1, one can in general see that the modified algorithm has increased performance compared to the original. Different settings give different results, and is highly dependent on the input video (shown in 5.1.4, Simulation 4). Firstly, the two different colour space will be compared before the prediction, and lastly the further possibilities.

Comparing the YEF and YUV colour space reveals that they have some similarities and some differences. Generally, the YEF is more energy efficient by reducing the average bitrate compared to YUV, but it “removes” black components. The tissues are slightly greyer, while the black areas contain red components. If ROI encoding and/or post-processing is applied, this reduction in red components could affect the result. Especially, if the post-processing analyses images by looking at reflected light intensity due to the variation of haemoglobin (Section 2.1.3 & 4.4). Simulating this effect is beyond the scope of the thesis, but would be something to consider. On the plus side, the average bitrate has decreased quite a bit by utilizing the YEF colour space instead of YUV. In other words, every transmitted frame uses less bits which saves energy consumption.

Less energy consumption is very favourable in the capsule as this allows for additional functionality and/or higher quality without changing the hardware. Quality increasing settings can be higher framerate or resolution. Additional functionality can be a second sensor (for instance, pH measuring sensor or ultrasound) or it can be error correcting codes (parity bit-/hamming codes, described in Section 3.3).

Error-correcting codes could be advantageous if a feedback loop is included since bit errors will propagate through the frames the same way BA DPCM does with pixel values (Section 2.4.3 & 4.2). Adapting the prediction coefficient showed this propagation of the bit errors in Simulation 3. Bit errors are in practice unavoidable when transmitting over a wireless channel, but errors can occur in compression schemes as well.

This is especially true with RLE coding (Section 2.6.3) which is very susceptible to bit errors since it has variable code-word length. Errors produced by the quantization will propagate through the system, resulting in a string off decoded errors. By reducing the number of quantized values that is coded together, the robustness would increase on the cost of higher bitrate (and higher energy consumption). It is expected that the propagating errors in the feedback loop would be less by applying the post-processing error correcting described in Section 3.3 [99].

Evaluating the feedback loop gives some important results. If the results from the first and second simulations is considered (Table 5.1 & Table 5.2), it can be seen that the quality measurements are approximately the same for adaptive and static prediction coefficient. However, the average bitrate per pixel has decreased, especially when the longer video is analysed instead of the short one. This is expected because adaption in the prediction coefficient will reduce the error and by that increase the number of successive equal values. Since consecutive values are grouped together in RLE encoding, this will reduce the average bitrate (Section 2.6.3).

Analysing the result in the third simulation (Table 5.3), on the other hand, show a decrease in performance as mentioned. In other words, adaptive prediction is dependent on the quality of the received frames. When the quality is sufficient for the adaptive prediction it seems to work better, especially with higher variation in the frame (e.g. not homogeneous frames).

Because all of the processing (finding the new prediction coefficient) is done at the decoder, additional calculations are possible and might increase the results even more. For instance, can only the non-corrupted pixels are used to update the coefficient, which can be combined with the error concealment and post-processing described earlier.

One of the important criteria in this application was to not increase the complexity and hardware. Since antennas are bidirectional in nature, this is maintained. The feedback loop can include many other parameters as well. As mentioned in Chapter 3, these can be variation in framerate, resolution, filter (in the MR part), or quantization steps. All of these will increase the energy consumption, but also increase the quality. The advantage with these is that they can be increased in some regions and decreased in others, giving an average energy consumption which can still be the same (or lower if chosen). This idea was investigated in a previous work (mentioned in Section 3.4), where the sampling rate in the MR was changed over different frames. As mentioned earlier (Section 4.3), this can also be done within the frames to create a ROI.

6.2 Part 2: ROI

Analysing the results from Chapter 5.2, the proposed method seems to be performing as expected, especially when studying the reconstructed videos. The ROI area includes all of the high frequency components, which are normally filtered out, but it also includes more on the horizontal and vertical axis of this area. This is exactly as expected, as described in Section 4.3.

The advantage with this kind of implementation is that no additional memory is required, and nearly no additional computations. The only calculation needed at the encoder is to calculate when to change the rate within a frame. This can be optimized and minimized compared to the current implementation, since the main goal in this thesis was to explore the idea.

Every calculation needed in order of locating the placement of the ROI area, is performed at the decoder, which then transmits the position back to the encoder in the feedback loop. As mentioned in the previous section (Section 6.1), it is uncertain how the removal of some red components in the YEF colour transform might affect finding the exact location of the region. Especially if this is based on the reflection of the light intensity by the protein haemoglobin.

Comparing the proposed method to other ROI encoding techniques, there are some differences. Other schemes often use DCT or wavelet transformation which requires additional memory and calculations. These methods extract the region or code only in the region with higher quality, depending on the specifications. The advantage with these methods is that no sections of the image that exceeds the ROI area will have to be coded with higher quality [100]. In other words, this reduces the required bitrate and the number of pixels to be coded in higher quality. This can reduce the energy consumption and the bandwidth compared to the proposed method, which can be a very limiting factor. However, it requires quite a bit more from the encoder, which is the main limiting factor in the WCE application. It is still uncertain how much the savings in computations together with the extra high quality pixels affects the total energy consumption, compared to the other methods.

As seen in the results from Chapter 5.2, there is some deviation from the expected evaluation measurements, compared to the results from Chapter 5.1. The scores were lower in two of the simulation videos, but higher in the third. When analysing the reconstructed videos, it was observable that the bit errors were reduced in the simulations, especially compared to the third simulation video. This might be due to the fact that more of the luminance component is kept (not filtered) along the whole column of the frame, resulting in larger variation in the values in that area. Since the prediction is done over a row, and quantized, fewer values will be rounded to zero. In other words, fewer bit errors will propagate through the scheme in the SR encoder.

On the other hand, in fast-changing frames, especially in bright areas, it seems that the ROI encoding might be having some difficulties. The axis seems to be corrupted in a few frames, which might explain the reduction in some of the objective scores. One possible reason might be that more of the high frequency is kept on the axis, which creates some corruption when the decimation/interpolation in the other direction is applied. This problem might be solved by optimizing the code some more. For instance, by improving the implemented interpolation method, since currently a built in function in MATLAB is used. This function utilizes a simple low-pass filter between the samples to create the new one in the interpolation process. It is believed that this filtering might be the reason for the errors, but the concern in this thesis was to simply test the proposed principle.

The main focus with the proposed system was the encoder process, and not the decoder. This seems to work in general, but it might need some improvements in the implementation process, especially concerning the energy evaluation. The average bit rate per pixel only calculates based on constant sampling rate over the whole frame, which then will be nearly identical or lower. Naturally, this isn’t correct since more pixels are transmitted with higher quality, so this will have to be tested when working with this further. For now, the complexity of the whole system will be examined further.

6.3 Complexity

The computational complexity of the proposed algorithm is very low. Especially the encoder is very energy efficient, since every additional computations is performed at the decoder. Compared to the original scheme, only a small module has to be added to the endoscope. This is the listening module, which will listen for incoming signal from the decoder for parameter update. Average bit per pixel has in this scheme been reduced by between 0.12-0.35 depending on which simulation video is used (Section 5.1), which will more than enough compensate for the additional listening module.

The feedback loop also allows for changes in the sampling rate which can reduce the bitrate more. This can either be performed in all colour components (YEF), or only reduce the subsampling in the chrominance components. Decreasing the sample rate will additionally reduce the bitrate, but will also decrease the quality. At the same time, the algorithm has the same complexity with this functionality, as the proposed.

The complexity of ROI encoding is also low, since no additional memory and nearly no computations is needed at the encoder. Additional computations are only needed in the higher quality areas, since these contains more pixels and aren't low-pass filtered. The extra pixels make the decimated frames larger than originally, but they increase the quality. Importantly, this is an optional functionality, so it can be active in only part of the capsule's whole journey through the GI tract. Every calculation in finding the region is done at the decoder, which then sends 4 integer values to the encoder; *x* and *y* position, as well as width and height.

The decoder has, on the other hand, higher complexity than before. More calculations are performed in both adapting the parameters and finding ROI region. Since the decoder has "unlimited" power compared to the encoder, this isn't considered a problem or within the scope of this thesis. As mentioned (Chapter 4), the focus will instead be at the encoder side, and the simulations performed to analyse the performance. However, these simulations can potentially have some sources of errors.

6.4 Error sources

As with every experiments and simulations, error sources are present resulting in deviation from the actual performance. The most obvious source of error or deviation, is the simulation videos. Performance evaluation is highly dependent on the input video, especially since every video is compressed (not in RAW format). This was shown in the fourth simulation (Section 5.1.4), where the recompression of the compressed simulation video had increase between 2-3.5dB in CPSNR, bitrate per pixel decreased by 0.11 and SSIM increased by 0.15. Naturally, the recompressed video wasn't of a higher quality than the input source, but the objective score would indicate that it was. Based on this principle, comparison between other encoding schemes for the WCE is quite difficult, since different simulation videos is used by different researchers. Objective performance measurements can only be compared when the simulation videos are identical[29]. However, a theoretical comparison can be analysed to try to get some pointers to how good performance is compared to other schemes, which will be presented in Section 6.5.

Similarly, the simulation videos can have negative impact compared to other simulation videos, as presented in 5.1.3. The cropped, recoded video (performed manually), resulted in a lot of bit errors from the quantization. This didn't happen at this degree in other simulations, which resulted in significantly reduction in performance. As mentioned (Section 6.1), RLE is very susceptible to bit errors, and with a feedback loop these might propagate. It is expected these wouldn't affect the performance if RAW images is used instead of converted videos, but should

[29] This was the case with the comparison with the performance measurements of original algorithm, shown in Section 5.1.

be taken into account when implementing. Especially since the channel will also introduce bit errors (Section 3.2), resulting in deviation from measured performance.

Deviation from actual implemented performance, can also be caused by the programming language used in the simulations. The simulations are performed with MATLAB, which has the advantage that it has very good matrix handling. Colour images can simply be regarded as three separate matrices (one for each colour channel), which can make MATLAB particularly well suited for image processing simulations. On the other hand, this language isn't directly implementable on a microchip, so translation into another programming language (for instance C or C++) is required. This might affect the performance results presented in this thesis, but generally, C/C++ is a more energy efficient programming language. This topic is discussed a bit further in Section 6.7.

6.5 Comparison

Comparing the performance in the proposed scheme with other schemes is quite difficult, without access to the identical simulation video or algorithm. A theoretical approach, on the other hand, is possible. The building blocks in the image compression scheme (Section 2.2.2), will be briefly evaluated by comparison with other methods.

Generally, wavelet transformation is slower and requires more computations than DCT or prediction, and has thus not gained full acceptance in most video coding standards (except in JPEG2000, which is used at digital cinemas) [15, 107]. DCT is more well established (used in JPEG [56]), but it still requires quite a bit of calculations (Section 2.4.1). This would require too much in an WCE application, but a simplified version called integer based DCT (iDCT[30]) has been used [50, 108, 109]. In this version of the transform, quantization might be redundant since this transform already constitutes loss. iDCT has been proven quite effective, but it does require additional temporary storages, since the frames are divided (tiled) into 8x8 blocks or similar block sizes. At low bitrates this will result in blocking artefacts in reconstructed frames, which is very noticeable and it gives poor quality [15]. Predictive coding prevents this blocking artefact, but it can be subject to degradation of compression performance when compressing at lossless or near-lossless [110]. A possible solution would be to apply some pre-processing, however, all of this results in an increase in complexity and energy consumption compared to the proposed algorithm. This is because most compression schemes for WCE usually has around 80-85% CR, while the proposed method has around 95-97% (depending on settings).

[30] Not to be confused with inverse DCT (IDCT).

Notice that some of these other schemes do have higher PSNR, but this is again very related to the source video used in simulations.

An advantage with the integer based DCT is that the quantization part can be omitted, partly since it is embedded inside the DCT transformation. The same principle can be done at entropy encoder by including a threshold together with the RLE. Neither results in lossless compression (still rounding at some point), but can reduce delay in the image compressor [109]. Additionally, this can be more advantageous if the compression scheme is to be near-lossless, but this is highly dependent on the available power capacity in the WCE.

Entropy coding is often combined with the different coding schemes (Section 2.6), which has different advantages and disadvantages. Generally, RLE is very efficient if there isn't a lot of variation in the coefficients, but it is susceptible to bit errors. Some combines RLE (zero coefficients) with Huffman or adaptive Golomb-Rice (AGR) (nonzero coefficients) [109, 111], similar to the JPEG standard [56], and some uses Lempel-Ziv (LZ) coding [112]. Common features among these methods are to minimize energy consumption, while maximizing quality. SR encoding has generally less storage and computational requirements than LZ, Huffman or AGR, but it is more compromised to bit errors. Especially, with the use of DZ quantizer which is used in the proposed scheme, the SR encoding won't necessary be the most suitable if this quantizer was omitted.

Avoiding the need for temporary storages or excessive computations will result in significant energy savings. However, most image compression standards don't focus on this. Generally, in many commercial applications, the encoder is the most complex part, and the decoder and available bit stream (bandwidth) is the limiting factor (for instance, mobile telephone system). This means that most standards don't have this high focus on very low power encoder savings, which is partly why it is an image- and not video compression scheme that is proposed (see next section).

6.6 Video Encoding

In this section, a brief comparison between the algorithm at hand, and video coding schemes will be presented.

In general, every video coding scheme consists of an image coder, video coder and audio coder. Obviously, audio coding isn't relevant in this application, and can be omitted (often optional). Video coding, on the other hand, is relevant. The main concept in a video coder compared to an image coder is to exploit correlation between frames instead of between pixels (described in

3.1.1.3), to reduce the required bitrate for each frame [15]. Because of the high correlation between the frames, prediction (Section 2.4.3) can be applied between the frames, especially in endoscopic images since the variation is very little. This is the same concept which is done in MPEG[31] standard, which utilizes both forward and backward prediction between frames. In addition, motion estimation and compensation (creating a motion vector (MV)) is done to estimate where areas or objects[32] in the frames will be in the next frame. The prediction is done over a group of pictures (GOP), which lowers the required bitrate, but this also has some disadvantages.

Prediction between frames introduces a delay equal to the GOP, i.e. more memory because of storage of the frames is required [15, 113]. Additionally, a lot more calculations is required, especially if motion estimation is performed. In total, this means that more storage (memory) is required, more components and more computations, i.e. more energy and physical hardware space is required. The capsule has limitations in all of these areas, but this would reduce the bitrate. In general, will there always be a trade-off between complexity (and bitrate) and quality. Increasing the quality will always also increase the complexity of the algorithm, and by that required energy/space.

Video encoding, instead of image encoder, is very dependent on the available hardware. If development in hardware (minimization in components, higher battery capacity, etc.) allows for a more complex algorithm, video encoding should be considered. Because of this high dependency on physical hardware, this is beyond the scope of the thesis but should be considered in future development.

6.7 Hardware Implementation

The algorithm is intended for implementation in an actuall WCE system, and the total energy consumption will highly depend on this process. The simulations are, as mentioned, done in a high level programming language. This has good matrix handling properties, but isn't very energy efficient or directly translatable. A translation is therefore necessary into an efficient language (for instance C++ or C#), which may affect the simulation results presented in this thesis.

In an implementation, the biggest change in the system is the additional listening module introduced. This allows for the system to listen for incoming signals from the RF antenna, which

[31] Movie Picture Expert Group (http://mpeg.chiariglione.org/)

[32] This can be done over a block (or tile), homogenous regions or a whole object.

can be used for updating the parameters. This means that no hardware change is required when implementing, because of the antennas bidirectional properties. However, energy efficient hardware has a great impact on the energy consumption, and current hardware should be evaluated since more energy efficient hardware constantly arrives.

In addition to the simulation algorithm, an effective channel coding may be required, together with a proper channel modelling (Section 3.2). A joint source-channel coding may be beneficial for low complexity modulation, and may be combined with SR encoding. Since SR encoding uses a 2-bit symbol representation, but varies in lengths, additional packing of the data stream can be combined with the channel encoding. Optimum channel coding (in complexity vs probability in transmission error) may depend on the chosen transmission method, as UWB and narrowband (NB) is different, and may have requirements in the channel modelling.

The specifics of the mentioned topics, is beyond the scope of this thesis but is highly relevant in a system implementation, and should be taken into consideration.

7 Conclusion

In this thesis an analysis of the algorithm used in the wireless video capsule endoscopy was investigated in order to find improvement potentials to increase the received image quality. Because there always is a trade-off between received quality and complexity in image encoders, a good solution was to reduce the average energy consumption for each transmission. With the use of the YEF colour space and adaptive prediction coefficients, the average energy consumption was reduced and the quality was slightly increased. Energy reduction was mainly measured in average bit per pixel, which decreased by between 0.12-0.35bpp depending on the input simulation video. The reduction in the average energy consumption will allow for higher resolution, frame rate or quality, or additional sensors on the WCE. The results were highly dependent on source video, so the results might differ in a real implementation.

A very low complexity ROI coding scheme was proposed for this application, which is based on adaptive sampling rate within the frames. This creates a kind of locally adaptive resolution (LAR), but don’t require any additional memory, or calculations compared to other ROI encoding schemes. Every calculation to locate the position of the regions are performed at the decoder, and the position is transmitted in a feedback loop to the encoder. This method doesn’t require any hardware changes (except for a listening module), since antennas are bidirectional in nature, resulting in an energy effective scheme.

The proposed ROI encoding scheme generally work as expected, but it seems that the decoding process has some troubles in the interpolation process outside the ROI area. This might be because of the low-pass filtering in the process doesn’t work optimally with variation in the sampling rate. The implementation process will require more investigation to make it work optimal in a real application. Especially in evaluating the increase in the energy consumption produced by the ROI encoding.

7.1 Future work

Even though some satisfying results have been given in this thesis, it is recommended to further work on this topic before implementing in a system for clinical trials. In the following section, some recommended suggestions from the author are presented, beyond the previously described hardware implementation (Section 6.7).

The main problem that haven’t been addressed in this thesis considers finding the particular ROI area. It has been mentioned (Section 2.1.3 & Chapter 4) a possible solution to finding the region, and this should be further investigated. A problem that could arise in this area, is the use of YEF colour transform instead of YUV. YEF requires less energy, and it gets better SNR, however, it also has a bit less SSIM and removes some of the red colour components. This could affect the ROI finding algorithm if it relies on reflected light from different parts of tissue. Another problem that could arise is if the received image is of too low quality (too much distortion or bit errors). Finding the region could be a difficult challenge then, and it should be investigated how reliable the algorithm is with different quality settings.

A weakness in the presented results are the actual simulation videos. These differ from the captured images from the CMOS sensor, and therefore the results will differ too. As shown in Table 5.4 (p. 69), compression of a compressed image/video will affect the result. It is recommended to perform further verification of the algorithm using RAW image data to explore the quality difference reliable. The results from this further verification will determine the total energy consumption, and thereby the allowed resolution/image quality parameters.

The implemented ROI scheme generally works as expected at the encoder, but it seems to struggle at the decoder. It should be investigated how the interpolation process can be optimized, specifically how the filtering process is performed in the built in function. Additionally, the energy evaluation, in the increase in bitrate should be modified to concerning adaptive sampling rate within the frame. At the moment it only evaluates as an average with constant sampling rate over the whole frame.

The ROI coding algorithm could be further developed by investigating if the quantization part in that region can be omitted (making the region near-lossless). It is uncertain how much this will affect the battery compared to the received quality in that area.

8 Bibliography

[1] GivenImaging. (2016, 6.6.2016). *Capsule Endoscopy: PillCam® SB 3* [Web Page]. Available: http://www.givenimaging.com/en-int/Innovative-Solutions/Capsule-Endoscopy/Pillcam-SB/PillCam-SB-3/Pages/default.aspx

[2] E. Scapa, H. Jacob, S. Lewkowicz, M. Migdal, D. Gat, A. Gluckhovski, *et al.*, "Initial experience of wireless-capsule endoscopy for evaluating occult gastrointestinal bleeding and suspected small bowel pathology," *The American Journal of Gastroenterology*, vol. 97, Issue: 11, pp. 2776-2779, Nov. 2002. Doi:10.1111/j.1572-0241.2002.07021.x

[3] N. Clearinghouse. (2013, 19.04.2016). The Digestive System and How It Works. *National Institue of Diabetes and Digestive and Kidney Diseases (NIDDK) NIH Publication No. 13–2681* 1-8. Available: http://www.niddk.nih.gov/health-information/health-topics/Anatomy/your-digestive-system/Documents/Digestive_System_508.pdf

[4] C. Ell, S. Remke, A. May, L. Helou, R. Henrich, and G. Mayer, "The First Prospective Controlled Trial Comparing Wireless Capsule Endoscopy with Push Enteroscopy in Chronic Gastrointestinal Bleeding," *Endoscopy*, vol. 34, Issue: 9, pp. 685-689, Sep. 2002. Doi:10.1055/s-2002-33446

[5] J. R. Saltzman, A. C. Travis, and R. S. Tilson. (2012, 06.06.2016). *American College of Gastroenterology (ACG): Small Bowel Bleeding and Capsule Endoscopy (3 ed.)* [Web Page]. Available: http://patients.gi.org/topics/capsule-endoscopy/

[6] B. Krans and S. Kim. (2015, 10.06.2016). *HealthLine - Endoscopy: Purpose, Procedure & Types* [Webpage]. Available: http://www.healthline.com/health/endoscopy#Overview1

[7] A. N. Kim, T. A. Ramstad, and I. Balasingham, "Very Low Complexity Low Rate Image Coding for the Wireless Endoscope," *ISABEL '11 Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, 2011.

[8] A. Perkis, "Compression," presented at the TTT4135: Multimedia signal processing, lecture, NTNU, Trondheim, 2016.

[9] R. Chávez-Santiago and I. Balasingham, "Ultrawideband Signals in Medicine [Life Sciences]," *IEEE Signal Processing Magazine*, vol. 31, Issue: 6, pp. 130-136, Oct. 2014. Doi:10.1109/MSP.2014.2340234

[10] D. Porcino and W. Hirt, "Ultra-Wideband Radio Technology: Potential And Challenges Ahead," *IEEE Communications Magazine*, vol. 41, Issue: 7, pp. 66-74, July 2003. Doi:10.1109/MCOM.2003.1215641

[11] X. Fang, H. Liu, G. Li, Q. Shao, and H. Li, "Wireless Power Transfer System for Capsule Endoscopy Based on Strongly Coupled Magnetic Resonance Theory," *2011 IEEE International Conference on Mechatronics and Automation*, pp. 232-236, Aug. 2011. Doi:10.1109/ICMA.2011.5985662

[12] T. Sun, X. Xie, G. Li, Y. Gu, Y. Deng, and Z. Wang, "A Two-Hop Wireless Power Transfer System With an Efficiency-Enhanced Power Receiver for Motion-Free Capsule Endoscopy Inspection," *IEEE Transactions on Biomedical Engineering*, vol. 59, Issue: 11, pp. 3247-3254, Nov. 2012. Doi:10.1109/TBME.2012.2206809

[13] E. V. Bø, "Trådløs Kapselendoskopi - Adaptiv koding for områdetilpasning," TTT4510 Fordypningsprosjekt, Fakultet for informasjonsteknologi, matematikk og elektroteknikk, NTNU, 2015.

[14] G. Pan and L. Wang, "Swallowable Wireless Capsule Endoscopy: Progress and Technical Challenges," *Gastroenterology Research and Practice*, vol. 2012, pp. 1-9, Oct. 2011. Doi:10.1155/2012/841691

[15] A. Perkis, "Visual Compression," presented at the TTT4135: Multimedia signal processing, lecture, NTNU, Trondheim, 2016.

[16] N. Clearinghouse. (2014, 10.06.2016). Crohn's Disease. *National Institue of Diabetes and Digestive and Kidney Diseases (NIDDK) NIH Publication No. 14–3410*, 1-16. Available: http://www.niddk.nih.gov/health-information/health-topics/digestive-diseases/crohns-disease/Documents/Crohns_508.pdf

[17] WebMD and J. Robinson. (2014, 21.04.2016). *Digestive Disorders Health Center: The Digestive System* [Web Page]. Available: http://www.webmd.com/digestive-disorders/digestive-system

[18] T. Taylor. (21.04.2016). *InnerBody: Digestive System* [Web Page]. Available: http://www.innerbody.com/image/digeov.html

[19] C. Blakemore and S. Jannett. (2001, 10.06.2016). *Encyclpoedia: Pharynx* [Web Page]. Available: http://www.encyclopedia.com/doc/1O128-pharynx.html

[20] T. Eisner and B. Raton. (2014, 21.04.2016). *MedlinePlus - U.S. National Library of Medicine: Digestive diseases* [Web Page]. Available: https://www.nlm.nih.gov/medlineplus/ency/article/007447.htm

[21] W. E. Whitehead. (2001, 21.04.2016). *IFFGD: Gastrointestinal Motility Disorders of the Small Intestine, Large Intestine, Rectum, and Pelvic Floor* [Web Page]. Available: http://www.iffgd.org/store/viewproduct/162

[22] N. J. Talley, "Functional gastrointestinal disorders as a public health problem," *Wiley Online Library,* vol. 10, Issue: Supplement s1, pp. 121-129, 8. April 2008. Doi:10.1111/j.1365-2982.2008.01097.x

[23] IFFGD. (2015, 21.04.2016). *Functional GI Disorders* [Web Page]. Available: http://www.iffgd.org/site/gi-disorders/functional-gi-disorders/

[24] IFFGD. (2016, 21.04.2016). *About GI Motility* [Web Page]. Available: http://www.aboutgimotility.org/

[25] IFFGD. (2015, 21.04.2016). *Motility Disorders* [Web Page]. Available: http://www.iffgd.org/site/gi-disorders/motility-disorders/

[26] IFFGD. (2014, 21.04.2016). *Other Disorders* [Web Page]. Available: http://www.iffgd.org/site/gi-disorders/other/

[27] J. Hopkins. (22.04.2016). *Health Library: Digestive Diagnostic Procedures* [Web Page]. Available: http://www.hopkinsmedicine.org/healthlibrary/conditions/digestive_disorders/digestive_diagnostic_procedures_85,P00364/

[28] K. Najarian and R. Splinter, "Introduction; Signal and Biomedical Signal Processing; Image Filtering, Enhancement and Restoration," in *Biomedical Signal and Image Processing*, 2nd ed Richmond, Virginia: CRC Press, 2012, pp. xxi-xxv; 3-13; 39-61.

[29] K. Najarian and R. Splinter, "X-Ray Imaging and Computed Tomography; Magnetic Resonance Imaging," in *Biomedical Signal and Image Processing*, 2nd ed Richmond, Virginia: CRC Press, 2012, pp. 261-307.

[30] NHS. (2015, 12.06.2016). *National Health Service (NHS): Laparoscopy (Keyhole Surgery)* [Web Page]. Available: http://www.nhs.uk/conditions/laparoscopy/Pages/Introduction.aspx

[31] NHS. (2014, 12.06.2016). *National Health Service (NHS): Endoscopy* [Web Page]. Available: http://www.nhs.uk/conditions/Endoscopy/Pages/Introduction.aspx

[32] WebMD and M. Ratini. (2015, 23.04.2016). *Digestive Disorders Health Center: Digestive Diseases and Endoscopy* [Web Page]. Available: http://www.webmd.com/digestive-disorders/digestive-diseases-endoscopy

[33] I. Balasingham and H. Fouladi, "Mini Project description: Compensating for zooming/panning and intensity variations in colonoscopy videos," presented at the TTT23: Biomedical Signal and Image Processing and Communications, Trondheim, 2015.

[34] ASH. (2016, 23.04.2016). *American Society of Hematology: Blood Basics* [Web Page]. Available: http://www.hematology.org/Patients/Basics/

[35] N. G. Roald, "Estimation of Vital Signs from Ambient-Light Non-Contact Photoplethysmography," Msc., Department of Electronics and Telecommunications, NTNU, Trondheim, 2013.

[36] Å. Rustand, "Ambient-light Photoplethysmography," Msc., Department of Electronics and Telecommunications, NTNU, Trondheim, 2012.

[37] J. D. Gibson, T. Berger, T. Lookabaugh, D. Lindbergh, and R. L. Baker, "Introduction to Data Compression; Lossless Source Coding; Quantization," in *Digital compression for multimedia, principles & standards*, 1st ed: Morgan Kaufmann publishers, 1998, pp. 1-138.

[38] A. S. Hornby, *Oxford Advanced Learner's Dictionary, 7th edition*. Oxford: Oxford University Press, 2005.

[39] B. Girod. (2000, 30.04.2016). *Stanford University - EE368B: Image and Video Compression* [Web Page]. Available: http://web.stanford.edu/class/ee368b/handouts.html

[40] B. Girod. (2000, 30.04.2016). Rate Distortion Theory. *EE368B: Image and Video Compression 4*. Available: http://web.stanford.edu/class/ee368b/Handouts/04-RateDistortionTheory.pdf

[41] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *The Bell System Technical Journal*, vol. 38, Issue: 3, pp. 611-656, 1959.

[42] B. Girod. (2000, 30.04.2016). Lossless Coding. *EE368B: Image and Video Compression 2*. Available: http://web.stanford.edu/class/ee368b/Handouts/02-LosslessCoding1.pdf

[43] D. Gunduz, E. Erkip, A. Goldsmith, and H. V. Poor, "Source and Channel Coding for Correlated Sources Over Multiuser Channels," *IEEE Transactions on Information Theory*, vol. 55, Issue: 9, pp. 3927-3944, September 2009.

[44] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 Still Image Coding System: An Overview " *IEEE Transactions on Consumer Electronics*, vol. 46, Issue: 4, pp. 1103-1127, November 2000. Doi:10.1109/30.920468

[45] J. Lodriguss. (2015, 05.05.2016). *Catching the Light: How Digital Cameras Work* [Web Page]. Available: http://www.astropix.com/HTML/I_ASTROP/HOW.HTM

[46] S. McHugh. (05.05.2016). *Cambridge in Colour: Digital Camera Sensors* [Web Page]. Available: http://www.cambridgeincolour.com/tutorials/camera-sensors.htm

[47] A. Ford and A. Roberts. (1998, 05.05.2016). Color Space Conversions. *Charles Poynton: Color Technology*, 1-31. Available: http://www.poynton.com/PDFs/coloureq.pdf

[48] A. Wong. (2011, 05.05.2016). *University of Waterloo: SYDE 575 -Image Processing* [Web Page]. Available: http://www.einfodaily.com/piTunez/syde575.htm

[49] D. Turgis and R. Puers, "Image compression in video radio transmission for capsule endoscopy," *Elsevier & Sensors and Actuators A: Physical*, vol. 123-124, pp. 129-136, Sep. 2005. Doi:10.1016/j.sna.2005.05.016

[50] A. Mostafa, T. H. Khan, S.-B. Ko, and K. Wahid, "Efficient color space-based compression scheme for endoscopic images," *IEEE - Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on,* pp. 83-86, 2012. Doi:10.1109/ISSPA.2012.6310670

[51] T. Khan and K. Wahid, "Design of a Lossless Image Compression System for Video Capsule Endoscopy and Its Performance in In-Vivo Trials," *Sensors*, vol. 14, Issue: 11, pp. 20779-20799, 2014. Doi:10.3390/s141120779

[52] V. Schwambach, S. Cleyet-Merle, A. Issard, and S. Mancini, "Image tiling for embedded applications with non-linear constraints," *IEEE: Design and Architectures for Signal and Image Processing (DASIP), 2015 Conference on* pp. 1-8, 23-25 Sept. 2015. Doi:10.1109/DASIP.2015.7367256

[53] R. Olanda, M. Pérez, and X. Benavent, "Tiling of the Wavelet Lowpass Subbands for Progressive Browsing of Images," *IEEE Signal Processing Letters*, vol. 13, Issue: 11, pp. 680-683, Nov. 2006. Doi:10.1109/LSP.2006.879468

[54] J.-R. Ohm, "Linear Systems and Transforms, Still Image Coding," in *Multimedia Communication Technology*. vol. 1, ed: Springer-Verlag Berlin Heidelberg, 2004, pp. 79-167, 509-549.

[55] J. G. Proakis and D. G. Manolakis, "The Discrete Fourier Transform: Its Properties and Applications; Multirate Digital Signal Processing," in *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed: Pearson Prentice Hall, 2007, pp. 449-501,750-790.

[56] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics,* vol. 38, Issue: 1, pp. xviii-xxxiv, feb. 1992. Doi:10.1109/30.125072

[57] A. Youssef. (2015, 10.05.2016). *The George Washington University: CS6351 Data Compression: Subband Coding & Wavelets* [Web Page]. Available: https://www.seas.gwu.edu/~ayoussef/cs6351/

[58] E. P. Simoncelli and E. H. Adelson. (1991, 10.05.2016). Chapter 4: Subband Transforms. *MIT Media Laboratory Vision and Modeling Technical Report #137,* 143-192. Available: http://persci.mit.edu/pub_pdfs/simoncelli_subband.pdf

[59] The-Crankshaft. (2011, 10.05.2015). *Mathematical Preliminaries (Image Processing) Part 1* [Web Page]. Available: http://what-when-how.com/embedded-image-processing-on-the-tms320c6000-dsp/mathematical-preliminaries-image-processing-part-1/

[60] K. Najarian and R. Splinter, "Wavelet Transform," in *Biomedical Signal and Image Processing*, 2nd ed Richmond, Virginia: CRC Press, 2012, pp. 79-100.

[61] P. Schniter, *An Introduction to Source-Coding: Quantization, DPCM, Transform Coding, and Sub-band Coding*. http://cnx.org/contents/b6387028-5ef5-4dbc-a7eb-ee9ea96d1b94@2.1: OpenStax CNX - Rice University, 2009.

[62] R. R. S. Tomar and K. Jain, "Lossless Image Compression Using Differential Pulse Code Modulation and its Application," *IEEE - Communication Systems and Network Technologies (CSNT), 2015 Fifth International Conference on* pp. 543-545, 4-6 April 2015. Doi:10.1109/CSNT.2015.192

[63] C. C. Cutler, "Differential Quantization of Communication Signals," United States Patent US2605361 A, July, 1952.

[64] J. G. Proakis and D. G. Manolakis, "Linear Prediction and Optimum Linear Filters," in *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed: Pearson Prentice Hall, 2007, pp. 823-879.

[65] J. G. Proakis and D. G. Manolakis, "Analog-to-Digital and Digital-to-Analog Conversion," in *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed: Pearson Prentice Hall, 2007, pp. 19-37.

[66] H. Gish and J. N. Pierce, "Asymptotically Efficient Quantizing," *IEEE Transactions on Information Theory,* vol. 14, Issue: 5, pp. 676-683, Sep 1968. Doi:10.1109/TIT.1968.1054193

[67] Y. Q. Shi. (2004, 20.05.2016). Chapter 2: Quantization. *New Jersey Institute of Technology - ECE 789: Digital Image Processing II* [Lecture]. 1-35. Available: https://web.njit.edu/~shi/courses/ECE789/ch2.pdf

[68] E. Modiano. (2009, 20.05.2016). Lecture 4: Quantization. *MIT OpenCourseWare - 16.36: Communication Systems Engineering* [Lecture]. 1-17. Available: http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-36-communication-systems-engineering-spring-2009/lecture-notes/MIT16_36s09_lec04.pdf

[69] G. J. Sullivan, "Efficient Scalar Quantization Of Exponential And Laplacian Random Variables " *IEEE Transactions on Information Theory*, vol. 42, Issue: 5, pp. 1365-1374, Sep. 1996. Doi:10.1109/18.532878

[70] D. Martinez and W. Yang, "A Robust Backward Adaptive Quantizer," *Neural Networks for Signal Processing [1995] V. Proceedings of the 1995 IEEE Workshop* pp. 531-540, 31. Aug. - 2. Sep. 1995. Doi:10.1109/NNSP.1995.514928

[71] C. A. Boyd, "Lecture 2: Classical Encryption," presented at the TTM4135: Information Security, lecture, NTNU, Trondheim, 2015.

[72] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic Coding For Data Compression," *Communications of the ACM*, vol. 30, Issue: 6, pp. 520-540, June 1987. Doi:10.1145/214762.214771

[73] S. Haykin, "Chapter 9: Fundamental Limits In Information Theory," in *Communication Systems*, 4th ed: John Wiley & Sons, Inc, 2001, pp. 567-617.

[74] C.-Y. Chen, Y.-T. Pai, and S.-J. Ruan, "Low Power Huffman Coding for High Performance Data Transmission," *IEEE - 2006 International Conference on Hybrid Information Technology (ICHIT'06)*, vol. 1, pp. 71-77, Nov 2006. Doi:10.1109/ICHIT.2006.253467

[75] A. H. Robinson and C. Cherry, "Results of a Prototype Television Bandwidth Compression Scheme," *Proceedings of the IEEE*, vol. 55, Issue: 3, pp. 356-364, March 1967. Doi:10.1109/PROC.1967.5493

[76] C. Tu, J. Liang, and T. D. Tran, "Adaptive Runlength Coding," *IEEE Signal Processing Letters*, vol. 10, Issue: 3, pp. 61-64, March 2003. Doi:10.1109/LSP.2002.807873

[77] M. J. Tsai, J. D. Villasenor, and F. Chen, "Stack-Run Image Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, Issue: 5, pp. 519-521, Oct 1996. Doi:10.1109/76.538934

[78] M.-J. Tsai, J. D. Villasenor, and F. Chen, "Stack-Run Coding for Low Bit Rate Image Communication " *Image Processing, 1996. Proceedings., International Conference on*, vol. 1, pp. 681-684, Sep 1996. Doi:10.1109/ICIP.1996.559590

[79] P. Raffy, C. Pépin, and R. M. Gray, "Robust Stack-Run Image Coding For Noisy Channels " *Signals, Systems, and Computers, 1999. Conference Record of the Thirty-Third Asilomar Conference on*, vol. 1, pp. 352-356, Oct. 1999. Doi:10.1109/ACSSC.1999.832351

[80] J. W. Schwartz and R. C. Barker, "Bit-Plane Encoding: A Technique for Source Encoding," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-2, Issue: 4, pp. 385-392, July 1966. Doi:10.1109/TAES.1966.4501787

[81] H. Kikuchi, K. Funahashi, and S. Muramatsu, "Simple Bit-Plane Coding for Lossless Image Compression and Extended Functionalities," *Picture Coding Symposium, 2009. PCS 2009* pp. 1-4, May 2009. Doi:10.1109/PCS.2009.5167351

[82] MayoClinic. (2015, 26.04.2016). *Tests and Procedures: Capsule endoscopy* [Web Page]. Available: http://www.mayoclinic.org/tests-procedures/capsule-endoscopy/basics/definition/prc-20012773

[83] GivenImaging, "Appendix A5: Technical Description," in *PillCam Capsule Endoscopy: User Manual, RAPID v8.0*, ed Hamburg: Given Imaging Ltd., 2013, pp. 175-205.

[84] O. Hinton. (2001, 30.05.2016). Chapter 9 – Multirate Digital Signal Processing. *"EEE305", "EEE801 Part A": Digital Signal Processing* [Lecture at University of Newcastle upon Tyne]. 9.1-9.8. Available: https://www.staff.ncl.ac.uk/oliver.hinton/eee305/Chapter9.pdf

[85] T. H. Khan and K. Wahid, "Lossless and Low-Power Image Compressor for Wireless Capsule Endoscopy," *VLSI Design*, vol. 1, pp. 1-12, May 2011. Doi:10.1155/2011/343787

[86] K. M. S. Thotahewa, J.-M. Redouté, and M. R. Yuce, "A UWB Wireless Capsule Endoscopy Device," *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* pp. 6977-6980, Aug. 2014. Doi:10.1109/EMBC.2014.6945233

[87] D. M. Pozar, "Introduction To Wireless Systems," in *Microwave and RF Design of Wireless Systems*, 1st ed: John Wiley & Sons, Inc, 2001, pp. 1-27.

[88] D. M. Pozar, "Antennas and Propagation for Wireless Systems," in *Microwave and RF Design of Wireless Systems*, 1st ed: John Wiley & Sons, Inc, 2001, pp. 111-147.

[89] S. Haykin, "Chapter 1: Random Processes," in *Communication Systems*, 4th ed: John Wiley & Sons, Inc, 2001, pp. 31-77.

[90] IEEE, "IEEE Standard for Safety Levels with Respect to Human Exposure to Radio Frequency Electromagnetic Fields, 3 kHz to 300 GHz," *IEEE Std C95.1-2005,* pp. 1-238, April 2006. Doi:10.1109/IEEESTD.2006.99501

[91] A. F. Molisch, "Chapter 5: Statistical Description of the Wireless Channel," in *Wireless Communications*, 1st ed: John Wiley & Sons, Inc., 2005, pp. 65-93.

[92] D. Tse and P. Viswanath, "The Wireless Channel," in *Fundamentals of Wireless Communication*, 1st ed: Cambridge University Press, 2005, pp. 10-48.

[93] K. Najarian and R. Splinter, "Electric Activities of the Cell," in *Biomedical Signal and Image Processing*, 2nd ed: CRC Press, 2012, pp. 155-169.

[94] M. R. Basar, M. F. B. A. Malek, K. M. Juni, M. I. M. Saleh, M. S. Idris, L. Mohamed, *et al.*, "The Use of a Human Body Model to Determine the Variation of Path Losses in the Human Body Channel in Wireless Capsule Endoscopy," *Progress In Electromagnetics Research (PIER),* vol. 133, pp. 495-513, 2013. Doi:10.2528/PIER12091203

[95] A. S. Bjørnevik, "Localization and Tracking of Intestinal Paths for Wireless Capsule Endoscopy," Msc., Department of Electronics and Telecommunications, Norwegian University of Science and Technology (NTNU), NTNU, 2015.

[96] G. Qvigstad, O. Fløttum, and H. L. Waldum. (2005, 01.06.2016). Kapselendoskopi – en ny metode for diagnostikk av sykdom i tynntarm. *Tidsskrift for Den norske legeforening 125(2),* 163-166. Available: http://tidsskriftet.no/pdf/pdf2005/163-6.pdf

[97] S. Haykin, "Chapter 10: Error-Control Coding," in *Communication Systems*, 4th ed: John Wiley & Sons, Inc, 2001, pp. 626-702.

[98] R. W. Hamming, "Error Detecting and Error Correcting Codes," *The Bell System Technical Journal*, vol. 29, Issue: 2, pp. 147-160, April 1950. Doi:10.1002/j.1538-7305.1950.tb00463.x

[99] A. N. Kim, E. J. Daling, T. A. Ramstad, and I. Balasingham, "Error Concealment and Post Processing for the Capsule Endoscope," *Proceedings of the 7th International Conference on Body Area Networks (BodyNets),* vol. 7, pp. 149-152, February 2012. Doi:10.4108/icst.bodynets.2012.249951

[100] O. Déforges, J. Ronsin, and L. Bedat, "The LAR Method as a New Scalable Region-Based Technique for Color Images Compression at Low Bit Rates " *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on* pp. 86-90, Jun. 2001. Doi:10.1109/ISPA.2001.938608

[101] F. Pasteau, M. Babel, O. Deforges, C. Strauss, and L. Bedat, "Chapter 3: Locally Adaptive Resolution (LAR) codec," in *Recent Advances in Signal Processing*, A. A. Zaher, Ed., 1 ed: InTech, 2009, pp. 37-48.

[102] O. Déforges and J. Ronsin, "Locally Adaptive Resolution Method for Progressive Still Image Coding," *Signal Processing and Its Applications, 1999. ISSPA '99. Proceedings of the Fifth International Symposium on,* vol. 2, pp. 825-829, Aug. 1999. Doi:10.1109/ISSPA.1999.815799

[103] O. Déforges and J. Ronsin, "Region of Interest Coding for Low Bit Rate Image Transmission," *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, vol. 1, pp. 107-110, July-Aug. 2000. Doi:10.1109/ICME.2000.869556

[104] E. U. Thodesen, I. Q. Marti, O. K. Melve, and Ø. A. Smith, "Compensation for zooming/panning and intensity variations in colonoscopy videos," TTT23: Biomedical signal and image processing and communications: Mini-project, IET, NTNU, 2015.

[105] A. D. Horchler. (2013, 05.06.2016). *QTWriter: Export QuickTime Movies with Matlab (1.1 ed.)* [Web Page]. Available: http://horchler.github.io/QTWriter/

[106] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, Issue: 4, pp. 600-612, April 2004. Doi:10.1109/TIP.2003.819861

[107] A. Perkis, "JPEG2000," presented at the TTT4136: Sound- & Image Processing, lecture, NTNU, Trondheim, 2013.

[108] A. Mostafa, T. Khan, and K. Wahid, "An Improved YEF-DCT based Compression Algorithm for Video Capsule Endoscopy," *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society* pp. 2452-2455, Aug. 2014. Doi:10.1109/EMBC.2014.6944118

[109] P. Turcza and M. Duplaga, "Low-Power Image Compression for Wireless Capsule Endoscopy," *2007 IEEE International Workshop on Imaging Systems and Techniques* pp. 1-4, May 2007. Doi:10.1109/IST.2007.379586

[110] X. Li, X. Xie, X. Chen, G. Li, L. Zhang, Z. Wang, *et al.*, "Design and Implementation of a Low Complexity Near-lossless Image Compression Method for Wireless Endoscopy Capsule System " *2007 IEEE International Symposium on Circuits and Systems* pp. 1321-1324, May 2007. Doi:10.1109/ISCAS.2007.378415

[111] P. Turcza and M. Duplaga, "Hardware-Efficient Low-Power Image Processing System for Wireless Capsule Endoscopy," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, Issue: 6, pp. 1046-1056, June 2013. Doi:10.1109/JBHI.2013.2266101

[112] K. W. S.-B. Ko, D. Teng, and V. Dimitrov, "Low-Area and Low-Power Video Compressor for Endoscopic Capsules," *Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on* pp. 507-510, May 2008. Doi:10.1109/CCECE.2008.4564586

[113] A. Perkis, "Characterisation of audiovisual information," presented at the TTT4135: Multimedia signal processing, lecture, NTNU, Trondheim, 2016.

Appendix A

A.1 Program list

- Matlab R2015b 64-bit (8.6.0.267246)
- Microsoft Word 2016 (Version 16.0.6001.1078)
- Endnote X7.5 (Bld 9325)
- MediaInfo GUI 0.7.78
- Avidemux 64-bit
- VLC media player (2.2.2 Weatherwax)

A.2 Simulation Videos Specifications

A.2.1 Video 1

General

Complete name	: E:\…\realseq3_c.avi
Source	: Ilangko Balasingham, NTNU
Format	: AVI
Format/Info	: Audio Video Interleave
File size	: 768 KiB
Duration	: 2s 125ms
Overall bit rate	: 2 960 Kbps

Video

ID	: 0
Format	: JPEG
Codec ID	: MJPG
Duration	: 2s 125ms
Bit rate	: 2 706 Kbps
Width	: 401 pixels
Height	: 241 pixels
Display aspect ratio	: 5:3
Frame rate	: 24.000 fps
Color space	: YUV
Chroma subsampling	: 4:2:0
Bit depth	: 8 bits
Compression mode	: Lossy
Bits/(Pixel*Frame)	: 1.167
Stream size	: 702 KiB (91%)

A.2.2 Video 2

General

Complete name	: E:\…\colon1.avi
Source	: Ilangko Balasingham, NTNU
Format	: AVI
Format/Info	: Audio Video Interleave
File size	: 83.7 MiB
Duration	: 1mn 37s

```
Overall bit rate                        : 7 171 Kbps
```

Video

```
ID                                      : 0
Format                                  : JPEG
Codec ID                                : MJPG
Duration                                : 1mn 37s
Bit rate                                : 7 166 Kbps
Width                                   : 768 pixels
Height                                  : 576 pixels
Original height                         : 1 152 pixels
Display aspect ratio                    : 4:3
Frame rate                              : 25.000 fps
Color space                             : YUV
Chroma subsampling                      : 4:2:2
Bit depth                               : 8 bits
Scan type                               : Interlaced
Compression mode                        : Lossy
Bits/(Pixel*Frame)                      : 0.648
Stream size                             : 83.7 MiB (100%)
```

A.2.3 Video 3

```
General
Complete name                           : E:\…\oesophagus_norm_crp_org.mp4
Source                                  :    Gastrolab    –    the    Gastrointestinal    Site:
http://www.gastrolab.net cropped with “Avidemux 64-bit”
Format                                  : MPEG-4
Format profile                          : Base Media / Version 2
Codec ID                                : mp42
File size                               : 6.19 MiB
Duration                                : 29s 640ms
Overall bit rate mode                   : Variable
Overall bit rate                        : 1 751 Kbps
Encoded date                            : UTC 2015-12-08 13:16:25
Tagged date                             : UTC 2015-12-08 13:16:31

Video
ID                                      : 1
Format                                  : AVC
Format/Info                             : Advanced Video Codec
Format profile                          : High@L2.1
Format settings, CABAC                  : Yes
Format settings, ReFrames               : 4 frames
Codec ID                                : avc1
Codec ID/Info                           : Advanced Video Coding
Duration                                : 29s 640ms
Duration_FirstFrame                     : 80ms
Bit rate                                : 1 622 Kbps
Width                                   : 430 pixels
Height                                  : 378 pixels
```

```
Display aspect ratio                     : 1.138
Frame rate mode                          : Constant
Frame rate                               : 25.000 fps
Color space                              : YUV
Chroma subsampling                       : 4:2:0
Bit depth                                : 8 bits
Scan type                                : Progressive
Bits/(Pixel*Frame)                       : 0.399
Stream size                              : 5.72 MiB (92%)
Writing library                          : x264 core 146 r2538 121396c
Encoding settings                        : cabac=1 / ref=3 / deblock=1:0:0 / analyse=0x3:0x133 /
me=hex / subme=7 / psy=1 / psy_rd=1.00:0.00 / mixed_ref=1 / me_range=16 / chroma_me=1
/ trellis=1 / 8x8dct=1 / cqm=0 / deadzone=21,11 / fast_pskip=1 / chroma_qp_offset=-2 /
threads=12 / lookahead_threads=2 / sliced_threads=0 / nr=0 / decimate=1 / interlaced=0 /
bluray_compat=0 / constrained_intra=0 / bframes=3 / b_pyramid=2 / b_adapt=1 / b_bias=0 /
direct=1 / weightb=1 / open_gop=0 / weightp=2 / keyint=250 / keyint_min=25 / scenecut=40 /
intra_refresh=0 / rc_lookahead=40 / rc=crf / mbtree=1 / crf=18.0 / qcomp=0.60 / qpmin=10 /
qpmax=51 / qpstep=4 / ip_ratio=1.40 / aq=1:1.00
```

Appendix B – MATLAB Code

B.1 Simulator

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
% Wireless Capsule Endocsopy simulator using GI tract videos
%
% Originally created by A. Kim 02.10.2011
% Last modified by E. Boe 15.06.2016
%
%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all; clc

Filename = 'colon1';
vidObj = VideoReader(strcat(Filename,'.avi'));
vidInfo = get(vidObj);
totFrames = vidInfo.Duration*vidInfo.FrameRate;


%Memory allocation
CPSNR = zeros(1,totFrames);
bit_rate = zeros(1,totFrames);
CR = zeros(1,totFrames);
pred_var = zeros(1,totFrames);
SSIM = zeros(1,totFrames);

%Define parameters
fig_name = 'colon1_adapt_YEF';
seq_num = 4;
colourTran = 'YEF';
a=0.9; %initial
p_coef = [a a -a^2]; % a=0.8, b=a, c=-a^2 [a b c]
sRate = [2 3];
deltas = [7, 6, 6];
quant_type = 'deadzone';
parameters = struct(...
    'colourTran',   colourTran,...
    'sRate',        sRate,...
    'p_coef',       p_coef,...
    'deltas',       deltas,...
    'quant_type',   quant_type,...
    'p_coef_mode',  'rows',...
    'enable_ROI',   1,...
    'ROI_pos',      [100 100 40 24],...
    'levels1',      0,...
    'levels2',      0,...
    'levels3',      0,...
    'frame_size',   0);

%create video write files
virecFileName =
sprintf('%s%d_downSample_%d%d_coef9_reconstr_%s.avi',Filename,seq_nu
m,sRate(1),sRate(2),colourTran);
virecObj = VideoWriter(virecFileName);
virecObj.FrameRate = 4;
```

```
open(virecObj);
vidifFileName=
sprintf('%s%d_downSample_%d%d_coef9_difference_%s.avi',Filename,seq_
num,sRate(1),sRate(2),colourTran);
vidifObj = VideoWriter(vidifFileName);
vidifObj.FrameRate = 4;
open(vidifObj);

if seq_num == 3
    rows = vidObj.Height-1;
    cols = vidObj.Width-1;
elseif seq_num == 4
    rows = vidObj.Height-6;
    cols = vidObj.Width-2;
else
    rows = vidObj.Height;
    cols = vidObj.Width;
end

%Run simulation
k=1;
while hasFrame(vidObj)
    display(['Frame ',int2str(k),' of ',num2str(totFrames)]);

    vidFrame = readFrame(vidObj);
    vidFrame = vidFrame(1:rows,1:cols,:);

    pred_var(k)=parameters.p_coef(1);

    %Encoder
    [sr123, parameters] = encoder(vidFrame, parameters);

    %Decoder
    [vidFrame_rec,parameters,component] = decoder(sr123,
parameters);%mean_RGB,

    %Performance evaluation
    [CPSNR(k),bit_rate(k),CR(k),SSIM(k)]=
performance_evaluation(vidFrame, vidFrame_rec, parameters);

    if parameters.enable_ROI
        writeVideo(virecObj, insertShape(vidFrame_rec,
'Rectangle',parameters.ROI_pos,'Color','Red','LineWidth',1));
        writeVideo(vidifObj, insertShape(vidFrame-vidFrame_rec,
'Rectangle',parameters.ROI_pos,'Color','Red','LineWidth',1));
    else
        writeVideo(virecObj, vidFrame_rec);
        writeVideo(vidifObj, vidFrame-vidFrame_rec);
    end
    if k==totFrames
        break;
    end
    k=k+1;
end

%end simulation program and close video files
```

```
close(virecObj);
close(vidifObj);

performFileName =
sprintf('%s%d_downSample_%d%d_coef9_performance_%s.mat',Filename,
seq_num,sRate(1),sRate(2),colourTran);
save(performFileName,'CPSNR', 'bit_rate', 'CR','SSIM');

display(['Average bit rate: ',num2str(mean(bit_rate))])
display(['Average compress ratio: ',num2str(mean(CR))])
display(['Average CPSNR: ',num2str(mean(CPSNR))])
display(['Average SSIM: ',num2str(mean(SSIM))])

%Plot results
figure(1)
set(gcf,'numbertitle','off','name',fig_name)
subplot(2,2,1),plot(1:totFrames,CPSNR,'b-*'),title('CPSNR
performance'),axis([0 50 35 40]);
axis tight;
ylabel('CPSNR in dB');
xlabel('Frame number');
grid on;

subplot(2,2,2), plot(1:totFrames,bit_rate,'b-d'),title('average bit
rate per pixel'),axis([0 50 0 1]);
grid on;
axis tight;
xlabel('Frame number');
ylabel('bit rate in bpp');

subplot(2,2,3), plot(1:totFrames,CR,'b-d'),title('average
compression ratio per pixel'),axis([0 50 0 1]);
grid on;
axis tight;
xlabel('Frame number');
ylabel('compression in precent');

subplot(2,2,4), plot(1:totFrames,SSIM,'b-o'),title('Average
structural similarity index measure'),axis([0 50 0 1]);
grid on;
axis tight;
xlabel('Frame number');
ylabel('SSIM');

figure(2)
set(gcf,'numbertitle','off','name',fig_name)
plot(1:totFrames,pred_var)
xlabel('Frame number');
ylabel('Prediction coefficient');
title('Variation in the prediction coefficient over the frames');
grid on;
axis tight;
```

B.2 Encoder

```
function varargout = encoder(vidFrame, parameters)

%Split frame into the different channels
R = double(vidFrame(:,:,1));
G = double(vidFrame(:,:,2));
B = double(vidFrame(:,:,3));

%Colour transformation
[c1_alt, c2_alt, c3_alt, mean_RGB] = en_colour_transform(R, G, B,
parameters);

%Multirate: lowpass & downsample with given sRate
if parameters.enable_ROI == 1
    [c1_MR, c2_MR, c3_MR] = en_lowpassDownsample_roi(c1_alt, c2_alt,
c3_alt, parameters);
else
    [c1_MR, c2_MR, c3_MR] =
en_lowpassDownsample(c1_alt,c2_alt,c3_alt, parameters);
end

%DPCM encoder
[levels1, levels2, levels3] = en_DPCM_encode(c1_MR,c2_MR,c3_MR,
parameters);

%encode quantization levels using SR run-length coder
sr123 = en_Frame_SRencode(levels1,levels2,levels3);

%update parameters:
parameters.mean_RGB = mean_RGB;
parameters.frame_size = size(vidFrame);
parameters.levels1 = levels1;
parameters.levels2 = levels2;
parameters.levels3 = levels3;

%send to decoder or to channelcoder
varargout{1} = sr123;
varargout{2} = parameters;

end
```

```
function [R_alt,G_alt,B_alt,mean_RGB] = en_colour_transform(R, G, B,
parameters)

if strcmp(parameters.colourTran,'YUV')
    transmat = [0.25 0.5 0.125; 0 -0.5 0.5; 0.5 -0.5 0];
elseif strcmp(parameters.colourTran,'YEF')
    transmat = [0.25 0.5 0.25; 0.125 -0.25 0.125; 0.125 0.125 -
0.25];
else
    error('encoder:unknownTransformation', 'Invalid colour
transformation method chosen')
end
```

```
R_alt = R*transmat(1,1)+G*transmat(1,2)+B*transmat(1,3);
G_alt = R*transmat(2,1)+G*transmat(2,2)+B*transmat(2,3)+128;
B_alt = R*transmat(3,1)+G*transmat(3,2)+B*transmat(3,3)+128;

mean_RGB = [mean(R_alt(:)) mean(G_alt(:)) mean(B_alt(:))];

R_alt = R_alt - mean_RGB(1);
G_alt = G_alt - mean_RGB(2);
B_alt = B_alt - mean_RGB(3);
End
```

```
function [c1_MR,c2_MR,c3_MR] = en_lowpassDownsample_roi(c1_alt,
c2_alt, c3_alt, parameters)
sRate = parameters.sRate;

x1 = parameters.ROI_pos(1);
y1 = parameters.ROI_pos(2);
w1 = parameters.ROI_pos(3);
h1 = parameters.ROI_pos(4);

size_c1 = size(c1_alt);
rowsC1 = size_c1(1);
colsC1 = size_c1(2);

tempR=zeros(rowsC1,(colsC1+w1)/sRate(1));
tempG=zeros(rowsC1,(colsC1+2*w1)/sRate(2));
tempB=zeros(rowsC1,(colsC1+2*w1)/sRate(2));

c1_MR = zeros((colsC1+w1)/sRate(1),((rowsC1+h1)/sRate(1)));
c2_MR = zeros((colsC1+2*w1)/sRate(2),(rowsC1+2*h1)/sRate(2));
c3_MR = zeros((colsC1+2*w1)/sRate(2),(rowsC1+2*h1)/sRate(2));

b = fir1(40,1/sRate(1));
Yfilt_c = filter2(b,c1_alt);

%Luminance component (Channel 1)
for i=1:rowsC1
    tempR(i,1:x1/sRate(1))=Yfilt_c(i,1:sRate(1):x1);
    tempR(i,x1/sRate(1):(x1/sRate(1)+w1))=c1_alt(i,x1:(x1+w1));

tempR(i,(x1/sRate(1)+w1):(colsC1+w1)/sRate(1))=Yfilt_c(i,(x1+w1):sRa
te(1):colsC1);
end
tempR = tempR';
for i = 1:(colsC1+w1)/sRate(1);
    c1_MR(i,1:y1/sRate(1)) = tempR(i,1:sRate(1):y1);
    c1_MR(i,y1/sRate(1):(y1/sRate(1)+h1)) = tempR(i,y1:(y1+h1));
    c1_MR(i,(y1/sRate(1)+h1):((rowsC1+h1)/sRate(1))) =
tempR(i,(y1+h1):sRate(1):rowsC1);
end

%Chrominance components (Channel 2 & 3)
for i=1:rowsC1
    tempG(i,1:(ceil(x1/sRate(2))))=c2_alt(i,1:sRate(2):x1);
tempG(i,(ceil(x1/sRate(2))):(ceil(x1/sRate(2))+w1))=c2_alt(i,x1:(x1+
w1));

tempG(i,(ceil(x1/sRate(2))+w1):(colsC1+2*w1)/sRate(2))=c2_alt(i,(x1+
w1):sRate(2):colsC1);

    tempB(i,1:(ceil(x1/sRate(2))))=c3_alt(i,1:sRate(2):x1);

tempB(i,(ceil(x1/sRate(2))):(ceil(x1/sRate(2))+w1))=c3_alt(i,x1:(x1+
w1));

tempB(i,(ceil(x1/sRate(2))+w1):(colsC1+2*w1)/sRate(2))=c3_alt(i,(x1+
w1):sRate(2):colsC1);
end
tempG=tempG';
tempB=tempB';
for i=1:(colsC1+2*w1)/sRate(2)
    c2_MR(i,1:(ceil(y1/sRate(2)))) = tempG(i,1:sRate(2):y1);
    c2_MR(i,(ceil(y1/sRate(2))):(ceil(y1/sRate(2))+h1)) =
tempG(i,y1:(y1+h1));
    c2_MR(i,(ceil(y1/sRate(2))+h1):(rowsC1+2*h1)/sRate(2)) =
tempG(i,(y1+h1):sRate(2):rowsC1);

    c3_MR(i,1:(ceil(y1/sRate(2)))) = tempB(i,1:sRate(2):y1);
    c3_MR(i,(ceil(y1/sRate(2))):(ceil(y1/sRate(2))+h1)) =
tempB(i,y1:(y1+h1));
    c3_MR(i,(ceil(y1/sRate(2))+h1):(rowsC1+2*h1)/sRate(2)) =
tempB(i,(y1+h1):sRate(2):rowsC1);
end

c1_MR = c1_MR';
c2_MR = c2_MR';
c3_MR = c3_MR';

end
```

```
%{
low-pass and down-sampling of original Y, U and V components.
sRate is the down-sampling rate , which also indicate the cut-off
frequency
of the low-pass filter
for U and V, only down-sampling is performed.
%}
function [c1_MR,c2_MR,c3_MR] = en_lowpassDownsample(c1_alt, c2_alt,
c3_alt, parameters)
sRate = parameters.sRate;
if sRate == 1
    c1_MR = c1_alt;
    c2_MR = c2_alt;
    c3_MR = c3_alt;
else
    %%% Filtering %%%
    b = fir1(40,1/sRate(1));
    Yfilt_c = filter2(b,c1_alt);
    %%% Downsample %%%
    Ydown_c = downsample(Yfilt_c,sRate(1));
    c1_MR = downsample(Ydown_c',sRate(1));
    c1_MR = c1_MR';

    c2_MR = downsample(c2_alt, sRate(2));
    c2_MR = downsample(c2_MR',sRate(2));
    c2_MR = c2_MR';

    c3_MR = downsample(c3_alt,sRate(2));
    c3_MR = downsample(c3_MR',sRate(2));
    c3_MR = c3_MR';
end
end
```

```
function [levelsY, levelsU, levelsV] = en_DPCM_encode(Y, U, V,
parameters)
p_coef = parameters.p_coef;
deltas = parameters.deltas;
quant_type = parameters.quant_type;

if nargin ~=4
    fprintf('ERROR: incorrect number of inputs!');
else
    levelsY = zeros(size(Y));
    levelsU = zeros(size(U));
    levelsV = zeros(size(V));

    Y_rec = levelsY;
    U_rec = levelsU;
    V_rec = levelsV;

    dY = levelsY;
    dU = levelsU;
    dV = levelsV;

    Yp = 0;
    Up = 0;
    Vp = 0;

    % quantization levels of the uniform mid-tread quantizer.
    deltaY = deltas(1);
    deltaU = deltas(2);
    deltaV = deltas(3);
    if strcmp(quant_type,'deadzone') == 1
        % quantization using deadzone quantizer. URURQ
        kY = 1;
        kU = 1;
        kV = 1;
        offsetY = 1-kY*deltaY*exp(-kY*deltaY)/(1-exp(-kY*deltaY));
        offsetU = 1-kU*deltaU*exp(-kU*deltaU)/(1-exp(-kU*deltaU));
        offsetV = 1-kV*deltaV*exp(-kV*deltaV)/(1-exp(-kV*deltaV));
    end
```

```
end
if (nargout ==3 || nargout == 6 || nargout == 9)
    %%% DPCM Y-component encode %%%
    for i = 2:size(Y,1)
        for j = 1:size(Y,2)
            dY(i,j) = Y(i,j)-Yp;
            if strcmp(quant_type,'uniquant')==1
                levelsY(i,j)=round(dY(i,j)/deltaY);
            else
                levelsY(i,j) =
sign(dY(i,j))*max(0,floor((abs(dY(i,j))-
(kY*deltaY)+offsetY)/(kY*deltaY)+1));
            end
            Y_rec(i,j) = Yp +levelsY(i,j)*deltaY;
            if j == size(Y,2)
                break;
            else
                Yp = p_coef(2)*Y_rec(i-
1,j+1)+p_coef(1)*Y_rec(i,j)+p_coef(3)*Y_rec(i-1,j);
            end
        end
    end
    %%% DPCM U,V-components encode %%%
    for i = 2:size(U,1)
        for j = 1:size(U,2)
            dU(i,j) = U(i,j)-Up;
            dV(i,j) = V(i,j)-Vp;
            if strcmp(quant_type,'uniquant')==1
                levelsU(i,j) = round(dU(i,j)/deltaU);
                levelsV(i,j) = round(dV(i,j)/deltaV);
            else
                levelsU(i,j) =
sign(dU(i,j))*max(0,floor((abs(dU(i,j))-
(kU*deltaU)+offsetU)/(kU*deltaU)+1));
                levelsV(i,j) =
sign(dV(i,j))*max(0,floor((abs(dV(i,j))-
(kV*deltaV)+offsetV)/(kV*deltaV)+1));
            end
            U_rec(i,j) = Up+levelsU(i,j)*deltaU;
            V_rec(i,j) = Vp+levelsV(i,j)*deltaV;
            if j == size(Y,2)
                break;
            else
                Up = p_coef(1)*U_rec(i,j);
                Vp = p_coef(1)*V_rec(i,j);
            end
        end
    end
else
    fprintf('ERROR: Incorrect number of outputs');
end

end
```

```
function srYUV = en_Frame_SRencode(levelsY, levelsU, levelsV)

rowsY = size(levelsY,1);
colsY = size(levelsY,2);
rowsU = size(levelsU,1);
colsU = size(levelsU,2);
segmentlength = [size(levelsY,2) size(levelsU,2)];

% skip the first row which is zero.
segment_nrY = ceil((rowsY-1)*colsY/segmentlength(1));
segment_nrU = ceil((rowsU-1)*colsU/segmentlength(2));

levelsY = levelsY(2:rowsY,:)';
levelsU = levelsU(2:rowsU,:)';
levelsV = levelsV(2:rowsU,:)';
srYUV = cell(3,max(segment_nrY,segment_nrU));

for i = 1:segment_nrY
        if i == segment_nrY && segment_nrY*segmentlength(1)> (rowsY-
1)*colsY
            last_pos = length(levelsY(:));
        else
            last_pos = i*segmentlength(1);
        end
        segmentY = levelsY((i-1)*segmentlength(1)+1:last_pos);
        tempY = en_SRencode(segmentY);
        srYUV(1,i) = {symbol2bits(tempY)};
end
for i = 1:segment_nrU
        if i == segment_nrU && segment_nrU*segmentlength(2)> (rowsU-
1)*colsU
            last_pos = length(levelsU(:));
        else
            last_pos = i*segmentlength(2);
        end
        segmentU = levelsU((i-1)*segmentlength(2)+1:last_pos);
        tempU = en_SRencode(segmentU);
        srYUV(2,i) = {symbol2bits(tempU)};

        segmentV = levelsV((i-1)*segmentlength(2)+1:last_pos);
        tempV = en_SRencode(segmentV);
        srYUV(3,i) = {symbol2bits(tempV)};
end
end
```

```
function Y = en_SRencode(x,FMT)
%pre allocate codeword size. -1 is dummy bit.
y = -ones(size(x));
sizey1 = 1; %starting position of the codeword
sizey2 = 0; %end position.
while ~isempty(x) % check if x has any value
    if isempty(find(x~=0, 1))% x has only zeros.
        RLbin_str = int2bit(length(x));
        % checking to see if run-length is 2^k-1
        if round(log2(length(x)+1))-log2(length(x)+1)~=0
            RLbin_str = RLbin_str(2:length(RLbin_str));
        end
        % mapping to '+','-'. where '+' is 3, '-' is 2.
        RLbin_str = RLbin_str +2;
        %determine end position of codeword
        sizey2 = sizey2+length(RLbin_str);
        % remove the encoded zeros.
        x = [];
        % assign to output and update starting position of codeword
        y(sizey1:sizey2)=RLbin_str;
        sizey1 = sizey2+1;
    else % x has both zero and nonzero elements
        if x(1)==0  % when first element is zero, code run length:
        % determine length of zeros, convert to binary.
        marker = length(x(1:find(x~=0, 1 )-1));
        RLbin_str = int2bit(marker);
        % checking to see if run-length is 2^k-1,if not remove MSB
        if round(log2(marker+1))-log2(marker+1)~=0
            RLbin_str = RLbin_str(2:length(RLbin_str));
        end
        %change 0, to -, 1 to +
        RLbin_str = RLbin_str +2;
        %update codeword start and end positions and output array
        sizey2 = sizey2+length(RLbin_str);
        y(sizey1:sizey2)=RLbin_str;
        sizey1 = sizey2+1;
        % remove the encoded zeros from input array
        x = x(find(x~=0,1):length(x));
        else % encode the first nonzero value
            % first increment the absolute value by 1. retain sign.
            nz_val = abs(x(1))+1;
            %NZbin_str = dec2bin(abs(nz_val));
            NZbin_str = int2bit(nz_val);
            % change the MSB into '+' or '-'
            if sign(x(1))>0
                NZbin_str(1)=3; % '+' is 3
            else
                NZbin_str(1)=2;% '-' is 2;
            end
            % update codeword positions and output array
            sizey2 = sizey2+length(NZbin_str);
            y(sizey1:sizey2)=NZbin_str;
            sizey1 = sizey2+1;
            % remove the nonzero value from input array
            if length(x)==1  % come to the last element
                x = [];
            else
                x = x(2:length(x));
            end
        end
    end
end
% remove the dummy bits -1;
y = y(y>=0);
```

```
if nargin==1 || strcmp(FMT,'double')==1
    Y = y;
elseif  strcmp(FMT, 'char')==1
    Y(y==3)  = '+';
    Y(y==2)  = '-';
    Y(y==0)  = '0';
    Y(y==1)  = '1';
else
    fprintf('ERROR: unknown output format.');
    Y = [];
end
end
```

```
% convert integer to bits. we use little Endian.
% N:    input integer. n: number of bits for output.
% n must be greater or equal to log2(N).
function bits = int2bit(N,n)
if nargin == 1
    if N == 0 || N == 1
        bits = rem(N,2);
    else
        n = floor(log2(N))+1;
        bits = zeros(1,n);
        for i = 1:n
            bits(i) = rem(N,2);
            N=(N-bits(i))/2 ;
        end
        bits = bits(end:-1:1);
    end
else
    bits = zeros(1,n);
    if N ==0  || N ==1
        bits(end) = rem(N,2);
    else
        for i = 1:n
            bits(i) = rem(N,2);
            N=(N-bits(i))/2 ;
        end
        bits = bits(end:-1:1);
    end
end
end
```

```
%{
converts symbols of +,-,0,1 into bits
uses 2 bits per symbol +: 11 -: 10 1: 01, 0:00
-1 is the dummy bit. it is transformed into [-1 -1];
%}
function bits = symbol2bits(symbols)
bits = zeros(2,size(symbols,2));
for i = 1: size(symbols,2)
    if symbols(i) =='+'|| symbols(i)== 3
        bits(:,i) = [1 1]';
    elseif symbols(i)== '-'|| symbols(i) == 2
        bits(:,i) = [1 0]';
    elseif symbols(i)=='1'|| symbols(i) == 1
        bits(:,i) = [0 1]';
    elseif symbols(i) == '0'|| symbols(i) == 0
        bits(:,i) = [0 0]';
    else
        bits(:,i) = ones(size(2,1))*symbols(i);%[-1 -1]';
    end
end
end
```

B.3 Decoder

```
function [vidFrame_rec,parameters,U_rec]= decoder(sr123, parameters)

%DPCM SR decode
[Y_rec, U_rec, V_rec, levelsY, levelsU, levelsV] =
de_DPCM_decode(sr123,parameters,0);

% MR - Lowpass and up sample
if parameters.enable_ROI == 1
    [co1_rec, co2_rec, co3_rec] = de_lowpassUpsample_roi(Y_rec,
U_rec, V_rec, parameters);
else
    [co1_rec, co2_rec, co3_rec] = de_lowpassUpsample(Y_rec, U_rec,
V_rec, parameters);
end
%inverse coulour transformation
[vidFrame_rec] = de_colour_transform
(co1_rec,co2_rec,co3_rec,parameters);

%Update parameters for next round
parameters = de_parameters_update(parameters, Y_rec, levelsY,
levelsU, levelsV);

end
```

```
function [Y_rec, U_rec, V_rec, levelsY, levelsU, levelsV] =
de_DPCM_decode(sr123,parameters,ROI_run)

if ROI_run
    levels1 = parameters.roi_levels1;
    levels2 = parameters.roi_levels2;
    levels3 = parameters.roi_levels3;
else
    levels1 = parameters.levels1;
    levels2 = parameters.levels2;
    levels3 = parameters.levels3;
end
    comp_size = [size(levels1); size(levels2); size(levels3)];
    Qrange = [max(levels1(:)) min(levels1(:)); max(levels2(:))
min(levels2(:)); max(levels3(:)) min(levels3(:))];
    segmentlength = [size(levels1,2) size(levels2,2)];
    %sr_YUV = sdec_input.sdec_bits;
    sr_YUV = sr123;
    deltas = parameters.deltas;
    p_coef = parameters.p_coef;
    maxY= Qrange(1,1);
    minY= Qrange(1,2);
    maxU= Qrange(2,1);
    minU= Qrange(2,2);
    maxV= Qrange(3,1);
    minV= Qrange(3,2);



   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % declare output                    %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    Y_rec = zeros(comp_size(1,:));
    U_rec = zeros(comp_size(2,:));
    V_rec = zeros(comp_size(3,:));

    colsY = size(Y_rec,2);
    rowsY = size(Y_rec,1);
    colsU = size(U_rec,2);
    rowsU = size(U_rec,1);

    %segmentlength = sdec_input.segmentlength;
    segment_nrY = ceil((rowsY-1)*colsY/segmentlength(1));
    segment_nrU = ceil((rowsU-1)*colsU/segmentlength(2));

    levelsY = zeros(1,colsY*(rowsY-1));
    levelsU = zeros(1,colsU*(rowsU-1));
    levelsV = zeros(1,colsU*(rowsU-1));

    Y_row = sr_YUV(1,:);
    U_row = sr_YUV(2,:);
    V_row = sr_YUV(3,:);


for k = 1:segment_nrY
    Y_symb = bits2symbol(Y_row{k});
    Y_symb = Y_symb(Y_symb>=0);
    if k == segment_nrY && segment_nrY*segmentlength(1)> (rowsY-
1)*colsY
        last_pos = length(levelsY(:));
        dlength = last_pos-(k-1)*segmentlength(1);
    else
        last_pos = k*segmentlength(1);
        dlength = segmentlength(1);
    end
    levelsY((k-1)*segmentlength(1)+1:last_pos) =
de_SRdecode(Y_symb,dlength);
end
for k = 1:segment_nrU
    U_symb = bits2symbol(U_row{k});
    V_symb = bits2symbol(V_row{k});
    U_symb = U_symb(U_symb>=0);
    V_symb = V_symb(V_symb>=0);
    if k == segment_nrU && segment_nrU*segmentlength(2)> (rowsU-
1)*colsU
        last_pos = length(levelsU(:));
        dlength = last_pos-(k-1)*segmentlength(2);
    else
        last_pos = k*segmentlength(2);
        dlength = segmentlength(2);
    end
    levelsU((k-1)*segmentlength(2)+1:last_pos) =
de_SRdecode(U_symb,dlength);
    levelsV((k-1)*segmentlength(2)+1:last_pos) =
de_SRdecode(V_symb,dlength);
end
    levelsY = [zeros(1,colsY) levelsY];
    levelsU = [zeros(1,colsU) levelsU];
    levelsV = [zeros(1,colsU) levelsV];
    levelsY = reshape(levelsY, colsY, rowsY);
    levelsU = reshape(levelsU, colsU, rowsU);
    levelsV = reshape(levelsV, colsU, rowsU);
    levelsY = levelsY';
    levelsU = levelsU';
    levelsV = levelsV';

    % constrain the max min values.
    levelsY(levelsY>maxY) = maxY;
    levelsY(levelsY<minY) = minY;
    levelsU(levelsU>maxU) = maxU;
    levelsU(levelsU<minU) = minU;
    levelsV(levelsV>maxV) = maxV;
    levelsV(levelsV<minV) = minV;

    % quantization levels of the uniform mid-tread quantizer.
    deltaY = deltas(1);
    deltaU = deltas(2);
    deltaV = deltas(3);
    Yp = 0;
    Up = 0;
    Vp = 0;

    % prediction coefficients for 2D DPCM, Y channel.
    a =p_coef(1);%.8;%1;
    b =p_coef(2);% .8;%a;
    c =p_coef(3);% -a^2;
    for i = 2:size(levelsY,1)
        for j = 1:size(levelsY,2)
            Y_rec(i,j) = Yp +levelsY(i,j)*deltaY;
            if j == size(levelsY,2)
                break;
            else
                Yp = b*Y_rec(i-1,j+1)+a*Y_rec(i,j)+c*Y_rec(i-1,j);
            end
        end
    end
    for i = 2:size(levelsU,1)
        for j = 1:size(levelsU,2)
            U_rec(i,j) = Up + levelsU(i,j)*deltaU;
            V_rec(i,j) = Vp + levelsV(i,j)*deltaV;
            if j == size(levelsU,2)
                break;
            else
                Up = a*U_rec(i,j);
                Vp = a*V_rec(i,j);
            end
        end
    end
end
```

```
function symbols = bits2symbol(bits,FMT)
% convert bits back to symbols
% bits dimension 2xL
L = size(bits,2);

if nargin == 1|| strcmp(FMT,'double')==1
    symbols = zeros(1,L);
    for l = 1:L
        if bits(:,l) ==[1 1]'
            symbols(l) = 3;
        elseif bits(:,l) ==[1 0]'
            symbols(l) = 2;
        elseif bits(:,l) ==[0 1]'
            symbols(l) = 1;
        elseif bits(:,l) ==[0 0]'
            symbols(:,l) = 0;
        else
            symbols(:,l) = -1;
        end
    end
else
    symbols = repmat('+',1,L);
    for l = 1:L
        if bits(:,l) ==[1 1]'
            symbols(l) = '+';
        elseif bits(:,l) ==[1 0]'
            symbols(l) = '-';
        elseif bits(:,l) ==[0 1]'
            symbols(l) = '1';
        elseif bits(:,l) ==[0 0]'
            symbols(:,l) = '0';
        else
            symbols(:,l) = 's';
        end
    end
end
```

```
function x = SRdecode(Y,output_length,FMT)
```

```
x = [];

if nargin == 0
    fprintf('ERROR: no input given');
elseif nargin ==1 || nargin == 2|| strcmp(FMT,'double')==1
    y = Y;
elseif strcmp(FMT, 'char')==1
    % converting string into to double.
    y(Y=='+') = 3;
    y(Y=='-') = 2;
    y(Y=='1') = 1;
    y(Y=='0') = 0;
else
    fprintf('ERROR: unknown input format.');
    y = [];
end

while ~isempty(y)
    if length(y)>1
        % find the '+','-', '1' and '0' positions
        PM_pos = find(y==3| y==2);
        OZ_pos = find(y==1|y==0);

        % y always starts with the symbol '+' or '-'

        if isempty(OZ_pos) % only runs of zeros.
            temp3 = y(1:length(y));
            y = [];
            temp3(temp3==3)=1;

            temp3(temp3==2)=0;

            numbr = bit2int([1 temp3]);

            % check if the length is 2^k-1

            if round(log2((numbr)+1))-log2((numbr)+1)==0
                rl = bit2int(temp3);
                rl_zeros = zeros(1,rl);
            else
                rl_zeros = zeros(1,numbr);
            end

                x = [x rl_zeros];

        elseif isempty(PM_pos) % only 1 or zeros. (not valid code
word)
            x = y;
            y = [];
        elseif ~isempty(OZ_pos) && OZ_pos(1)==2 % first codeword is
none-zero value

            % check the sign of the none-zero value.
            if y(PM_pos(1))==3
               nz_sign = 1;
            elseif y(PM_pos(1))==2
```

```
                nz_sign = -1;
            end


            %convert the binary into decimal.
            % remove the codeword from y

            if length(PM_pos)>1
                temp = [1 y(2:PM_pos(2)-1)];
                y = y(PM_pos(2):length(y));

            else% no other valide codeword in the input.
                temp = [1 y(2:length(y))];
                y = [];
            end

            x = [x nz_sign*(bit2int(temp)-1)];

        elseif OZ_pos(1) ==1 %first codword is 1/0. decode as 1/0.
                x = [x y(1)];

                % remove codeword.
                y = y(2:length(y));

        elseif OZ_pos(1)>2% first code word is run length of zeros.
                        %if ~isempty(OZ_pos)
                temp3 = y(1:OZ_pos(1)-2);% remember the nz val has
MSB.
                y = y(OZ_pos(1)-1:length(y));

                temp3(temp3==3)=1;
                temp3(temp3==2)=0;
                numbr = bit2int([1 temp3]);

                if round(log2((numbr)+1))-log2((numbr)+1)==0 % if
the length is 2^k-1
                    rl = bit2int(temp3); % then all bits are there
                    rl_zeros = zeros(1,rl);
                else % if not, need the one with added bits.
                    rl_zeros = zeros(1,numbr);
                end


                x = [x rl_zeros];
        end

    elseif length(y)==1
        if y == 3 || y == 0% here y = +, then only 1 zero is left.
            x = [x 0];
        elseif y == 2 % here y = -, two zeros are left.
            x = [x 0 0];
        elseif y == 1
            x = [x 1];

        end
            y = [];
    end
end

%check to see if x is at the right length.
if nargin >=2
    if output_length > length(x)
        x = [x zeros(1,output_length-length(x))];
    else
        x = x(1:output_length);
    end
end
```

```
function N =bit2int(bits)
% convert bits to integers.

n = length(bits)-1;
w =2.^(n:-1:0);
N = sum(bits.*w);
End
```

```
function [co1_rec, co2_rec, co3_rec] =
de_lowpassUpsample_roi(c1_down, c2_down, c3_down, parameters)
sRate = parameters.sRate;
x1 = parameters.ROI_pos(1);
y1 = parameters.ROI_pos(2);
w1 = parameters.ROI_pos(3);
h1 = parameters.ROI_pos(4);

sizeUo = size(c2_down);
rowsU = sizeUo(1);
colsU = sizeUo(2);
sizeYo = size(c1_down);
rowsY = sizeYo(1,1);
colsY = sizeYo(1,2);
temp1 = zeros(rowsY,(sRate(1)*colsY-w1));
temp2 = zeros(rowsU,(sRate(2)*colsU-2*w1));
temp3 = zeros(rowsU,(sRate(2)*colsU-2*w1));
c1_rec = zeros((sRate(1)*colsY-w1),(sRate(1)*rowsY-h1));
c2_rec = zeros((sRate(2)*colsU-2*w1),(sRate(2)*rowsU-2*h1));
c3_rec = zeros((sRate(2)*colsU-2*w1),(sRate(2)*rowsU-2*h1));
c1_down = c1_down(2:rowsY,1:colsY);
c1_down = [c1_down;.1*c1_down(end,1:colsY)];

c2_down = c2_down(2:rowsU,1:colsU);
c2_down = [c2_down; zeros(1,colsU)];

%Luminance component (Channel 1)
for i = 1:rowsY
    temp1(i,1:x1) =
interp(c1_down(i,1:(x1/sRate(1))),sRate(1),10,.5);
    temp1(i,x1:(x1+w1)) = c1_down(i,(x1/sRate(1)):(x1/sRate(1)+w1));
    temp1(i,(x1+w1+1):(sRate(1)*colsY-w1)) =
interp(c1_down(i,(x1/sRate(1)+w1+1):colsY),sRate(1),10,.5);
```

```
end
temp1 = temp1';
for i = 1:(sRate(1)*colsY-w1)
    c1_rec(i,1:y1) =
interp(temp1(i,1:(y1/sRate(1))),sRate(1),10,.5);
    c1_rec(i,y1:(y1+h1)) = temp1(i,(y1/sRate(1)):(y1/sRate(1)+h1));
    c1_rec(i,(y1+h1+1):(sRate(1)*rowsY-h1)) =
interp(temp1(i,(y1/sRate(1)+h1+1):rowsY),sRate(1),10,.5);
end

%Chrominance components (Channel 2 & 3)
for i = 1:rowsU
    temp2(i,1:(x1-1)) = interp(c2_down(i,1:(floor(x1/sRate(2)))),
sRate(2));
    temp2(i,x1:(x1+w1)) =
c2_down(i,(floor(x1/sRate(2))+1):(floor(x1/sRate(2))+w1+1));
    temp2(i,(x1+w1):(sRate(2)*colsU-2*w1)) =
interp(c2_down(i,(floor(x1/sRate(2))+w1+1):colsU), sRate(2));

    temp3(i,1:(x1-1)) = interp(c3_down(i,1:(floor(x1/sRate(2)))),
sRate(2));
    temp3(i,x1:(x1+w1)) =
c3_down(i,(floor(x1/sRate(2))+1):(floor(x1/sRate(2))+w1+1));
    temp3(i,(x1+w1):(sRate(2)*colsU-2*w1)) =
interp(c3_down(i,(floor(x1/sRate(2))+w1+1):colsU), sRate(2));
end
temp2 = temp2';
temp3 = temp3';
for i = 1:(sRate(2)*colsU-2*w1)
    c2_rec(i,1:(y1-1)) =
interp(temp2(i,1:(floor(y1/sRate(2)))),sRate(2));
    c2_rec(i,y1:(y1+h1)) =
temp2(i,(floor(y1/sRate(2))+1):(floor(y1/sRate(2))+h1+1));
    c2_rec(i,(y1+h1):(sRate(2)*rowsU-2*h1)) =
interp(temp2(i,(floor(y1/sRate(2))+h1+1):rowsU),sRate(2));

    c3_rec(i,1:(y1-1)) =
interp(temp3(i,1:(floor(y1/sRate(2)))),sRate(2));
    c3_rec(i,y1:(y1+h1)) =
temp3(i,(floor(y1/sRate(2))+1):(floor(y1/sRate(2))+h1+1));
    c3_rec(i,(y1+h1):(sRate(2)*rowsU-2*h1)) =
interp(temp3(i,(floor(y1/sRate(2))+h1+1):rowsU),sRate(2));
end

c1_rec = c1_rec';
c2_rec = c2_rec';
c3_rec = c3_rec';
co1_rec = wiener2(c1_rec,[3,3]);
co2_rec = wiener2(c2_rec,[3,3]);
co3_rec = wiener2(c3_rec,[3,3]);
end
```

```
function [co1_rec, co2_rec, co3_rec] = de_lowpassUpsample(Y_down,
U_down, V_down, parameters)
```

```
    sRate = parameters.sRate;

    sizeUo = size(U_down);
    rowsU = sizeUo(1);
    colsU = sizeUo(2);
    sizeYo = size(Y_down);
    rowsY = sizeYo(1,1);
    colsY = sizeYo(1,2);

    Y_down = Y_down(2:rowsY,1:colsY);
    Y_down = [Y_down;.1*Y_down(end,1:colsY)];

    U_down = U_down(2:rowsU,1:colsU);
    U_down = [U_down; zeros(1,colsU)];

    for i = 1:rowsY
        Yup_row(i,:) = interp(Y_down(i,:),sRate(1),10,.5);
    end
    Yup_rowtrans = Yup_row';
    for i = 1:colsY*sRate(1)
        Y2(i,:) = interp(Yup_rowtrans(i,:),sRate(1),10,.5);
    end

    for i = 1:rowsU
        Uup_row(i,:) = interp(U_down(i,:), sRate(2));
        Vup_row(i,:) = interp(V_down(i,:), sRate(2));
    end
    Uup_rowtrans = Uup_row';
    Vup_rowtrans = Vup_row';
    for i = 1:colsU*sRate(2)
        U2(i,:) = interp(Uup_rowtrans(i,:),sRate(2));
        V2(i,:) = interp(Vup_rowtrans(i,:),sRate(2));
    end

    Y2 = Y2';
    U2 = U2';
    V2 = V2';
    co1_rec = wiener2(Y2,[3,3]);
    co2_rec = wiener2(U2,[3,3]);
    co3_rec = wiener2(V2,[3,3]);
end
```

```
function [vidFrame_rec] = de_colour_transform
(co1_rec,co2_rec,co3_rec,parameters)

mean_RGB = parameters.mean_RGB;

frame_size = size(co1_rec);
rows = frame_size(1);
cols = frame_size(2);
vidFrame_rec = uint8(zeros(frame_size));

if strcmp(parameters.colourTran,'YUV')
    transmat = [0.25 0.5 0.125; 0 -0.5 0.5; 0.5 -0.5 0];
```

```
elseif strcmp(parameters.colourTran,'YEF')
    transmat = [0.25 0.5 0.25; 0.125 -0.25 0.125; 0.125 0.125 -
0.25];
else
    error('encoder:unknownTransformation', 'Invalid colour
transformation method chosen')
end

mean_co1 = mean_RGB(1);
mean_co2 = mean_RGB(2);
mean_co3 = mean_RGB(3);
if diff(parameters.sRate) ~= 0  || max(parameters.sRate)>1
     co1_rec = [co1_rec(1,1:cols); co1_rec(1:rows-1,1:cols)];
     co2_rec = [co2_rec(1,1:cols); co2_rec(1:rows-1,1:cols)];
     co3_rec = [co3_rec(1,1:cols); co3_rec(1:rows-1,1:cols)];
end
invtransmat = inv(transmat);

vidFrame_rec(:,:,1) =
uint8(invtransmat(1,1)*(co1_rec+mean_co1)+invtransmat(1,2)*(co2_rec+
mean_co2-128) + invtransmat(1,3)*(co3_rec+mean_co3-128));
vidFrame_rec(:,:,2) =
uint8(invtransmat(2,1)*(co1_rec+mean_co1)+invtransmat(2,2)*(co2_rec+
mean_co2-128) + invtransmat(2,3)*(co3_rec+mean_co3-128));
vidFrame_rec(:,:,3) =
uint8(invtransmat(3,1)*(co1_rec+mean_co1)+invtransmat(3,2)*(co2_rec+
mean_co2-128) + invtransmat(3,3)*(co3_rec+mean_co3-128));
end
```

```
function parameters = de_parameters_update(parameters, component,
levelsY, levelsU, levelsV)
component( ~any(component,2), : ) = [];  %rows
component( :, ~any(component,1) ) = [];  %columns
if strcmp(parameters.p_coef_mode, 'colums')
    for i=1:length(component(1,:))
        autcorr_c(i,:)=autocorr(component(:,i)');
    end

    mean_c = zeros(1,length(autcorr_c(1,:)));

    for i=1:length(autcorr_c(1,:))
        mean_c(i)=mean(autcorr_c(1:end,i));
    end
    a = mean_c(2);
elseif strcmp(parameters.p_coef_mode, 'rows')
    for j=1:length(component(:,1))
        autcorr_r(j,:)=autocorr(component(j,:));
    end

    mean_r = zeros(1,length(autcorr_r(1,:)));

    for i=1:length(autcorr_r(1,:))
        mean_r(i)=mean(autcorr_r(2:end,i));
    end
    a = mean_r(2);
end

parameters.p_coef = [a a -a^2];

parameters.levels1 = levelsY;
parameters.levels2 = levelsU;
parameters.levels3 = levelsV;
end
```

B.4 Performance Evaluate

```
function [CPSNR,rate_bpp,CR,SSIM]=
performance_evaluation(vidFrame,vidFrame_rec,parameters)
sRate = parameters.sRate;

%encoder - decoder
[R_co,G_co,B_co,mean_RGB] =
en_colour_transform(double(vidFrame(:,:,1)),double(vidFrame(:,:,2)),
double(vidFrame(:,:,3)),parameters);

if parameters.enable_ROI == 1
    [colour1,colour2,colour3] =
en_lowpassDownsample_roi(R_co,G_co,B_co, parameters);
else
    [colour1,colour2,colour3] = en_lowpassDownsample(R_co,G_co,B_co,
parameters);
end

[levels1, levels2, levels3] =
en_DPCM_encode(colour1,colour2,colour3,parameters);

[rate_sr1,rate_sr2,rate_sr3] =
pe_Frame_SRencode(levels1,levels2,levels3);

%--- Performance evaluation ---
%}
%%% Average rate per pixel %%%
mean_rY = mean(rate_sr1)/sRate(1)^2;
mean_rU = mean(rate_sr2)/sRate(2)^2;
mean_rV = mean(rate_sr3)/sRate(2)^2;
rate_bpp = mean_rY+mean_rU+mean_rV;

%%% PSNR performance %%%
CPSNR= eval_CPSNR(vidFrame,vidFrame_rec,parameters);

%%% SSIM %%%
SSIM = ssim(vidFrame_rec,vidFrame);

%%% Calculate CR %%%
CR = (1-rate_bpp/24)*100;
End
```

```
function varargout = pe_Frame_SRencode(varargin)
if nargin ~=3 && nargin~=4
```

```
    fprintf('ERROR: incorrect number of inputs!');
else
    levelsY = varargin{1};
    levelsU = varargin{2};
    levelsV = varargin{3};
    segmentlength = [size(levelsY,2) size(levelsU,2)];
    rowsY = size(levelsY,1);
    colsY = size(levelsY,2);
    rowsU = size(levelsU,1);
    colsU = size(levelsU,2);
    segment_nrY = ceil((rowsY-1)*colsY/segmentlength(1));
    segment_nrU = ceil((rowsU-1)*colsU/segmentlength(2));
end
levelsY = levelsY(2:rowsY,:)';
levelsU = levelsU(2:rowsU,:)';
levelsV = levelsV(2:rowsU,:)';
rate_srY = zeros(1,segment_nrY);
rate_srU = zeros(1,segment_nrU);
rate_srV = zeros(1,segment_nrU);

for i = 1:segment_nrY
        if i == segment_nrY && segment_nrY*segmentlength(1)> (rowsY-
1)*colsY
            last_pos = length(levelsY(:));
        else
            last_pos = i*segmentlength(1);
        end
        segmentY = levelsY((i-1)*segmentlength(1)+1:last_pos);
        tempY = en_SRencode(segmentY);
        rate_srY(i) =2*length(tempY)/segmentlength(1); % 2bit is
used for coding each symbol.
end
for i = 1:segment_nrU
        if i == segment_nrU && segment_nrU*segmentlength(2)> (rowsU-
1)*colsU
            last_pos = length(levelsU(:));
        else
            last_pos = i*segmentlength(2);
        end
        segmentU = levelsU((i-1)*segmentlength(2)+1:last_pos);
        tempU = en_SRencode(segmentU);
        rate_srU(i) =2*length(tempU)/segmentlength(2); % 2bit is
used for coding each symbol.
        segmentV = levelsV((i-1)*segmentlength(2)+1:last_pos);
        tempV = en_SRencode(segmentV);
        rate_srV(i) =2*length(tempV)/segmentlength(2); % 2bit is
used for coding each symbol.
end
varargout(1) = {rate_srY};
varargout(2) = {rate_srU};
varargout(3) = {rate_srV;
end
```

```
function CPSNR = eval_CPSNR(pic,pic_rec,parameters)
```

```
% declare constants
rows = size(pic,1);
cols = size(pic,2);
%split frames
co1 = double(pic(:,:,1));
co2 = double(pic(:,:,2));
co3 = double(pic(:,:,3));
co1_rec = double(pic_rec(:,:,1));
co2_rec = double(pic_rec(:,:,2));
co3_rec = double(pic_rec(:,:,3));

if strcmp(parameters.colourTran,'YUV')
    transmat = [0.25 0.5 0.125; 0 -0.5 0.5; 0.5 -0.5 0];
elseif strcmp(parameters.colourTran,'YEF')
    transmat = [0.25 0.5 0.25; 0.125 -0.25 0.125; 0.125 0.125 -
0.25];
else
    error('encoder:unknownTransformation', 'Invalid colour
transformation method chosen')
end
co1_o = co1*transmat(1,1)+co2*transmat(1,2)+co3*transmat(1,3);
co2_o = co1*transmat(2,1)+co2*transmat(2,2)+co3*transmat(2,3)+128;
co3_o = co1*transmat(3,1)+co2*transmat(3,2)+co3*transmat(3,3)+128;
co1_rec_o =
co1_rec*transmat(1,1)+co2_rec*transmat(1,2)+co3_rec*transmat(1,3);
co2_rec_o =
co1_rec*transmat(2,1)+co2_rec*transmat(2,2)+co3_rec*transmat(2,3)+12
8;
co3_rec_o =
co1_rec*transmat(3,1)+co2_rec*transmat(3,2)+co3_rec*transmat(3,3)+12
8;

mean_co1_o = mean(co1_o(:));
mean_co2_o = mean(co2_o(:));
mean_co3_o = mean(co3_o(:));
mean_co1_rec_o = mean(co1_rec_o(:));
mean_co2_rec_o = mean(co2_rec_o(:));
mean_co3_rec_o = mean(co3_rec_o(:));

co1_o = co1_o-mean_co1_o;
co2_o = co2_o-mean_co2_o;
co3_o = co3_o-mean_co3_o;
co1_rec_o = co1_rec_o-mean_co1_rec_o;
co2_rec_o = co2_rec_o-mean_co2_rec_o;
co3_rec_o = co3_rec_o-mean_co3_rec_o;

if diff(parameters.sRate) ~= 0  || max(parameters.sRate)>1
     co1_rec_o = [co1_rec_o(1,1:cols); co1_rec_o(1:rows-1,1:cols)];
     co2_rec_o = [co2_rec_o(1,1:cols); co2_rec_o(1:rows-1,1:cols)];
     co3_rec_o = [co3_rec_o(1,1:cols); co3_rec_o(1:rows-1,1:cols)];
end
df_1 = co1_o(2:rows,2:cols)-co1_rec_o(2:rows,2:cols);
df_2 = co2_o(2:rows,2:cols)-co2_rec_o(2:rows,2:cols);
df_3 = co3_o(2:rows,2:cols)-co3_rec_o(2:rows,2:cols);
```

```
%composite PSNR
CPSNR =
10*log10((255^2*3)/(var(df_1(:))+var(df_2(:))+var(df_3(:))));
End
```