Embedded Video Compression

Compression for High Dynamic Range Video

Leonardo Bergesio

Supervisor: Onno Eerenberg

Year: 2009-2010

Authors' address data: L. Bergesio, HTC41

:

Abstract

Modern digital TV-systems encompass digital demodulation and required audiovisual signal processing incorporating an unified-memory architecture. The bottleneck of such a System-on-Chip (SoC) is the connection to the external memory. As the video signal consumes the majority of the involved bandwidth, embedded video compression allows bandwidth reduction, enabling cost effective systems. The embedded video compression must be visual lossless and cost effective. This requires a high efficient de-correlation step to limit the amount of quantization, thus avoiding a decrease of picture quality. In this thesis, video characterization has been conducted, resulting in novel video de-correlation concepts. Based on these concepts, a DCT-based codec exploiting symmetrical features, structure repetitions and prediction techniques in the DCT transform domain has been developed. From the results it can be concluded, that symmetrical features turn out to be an efficient method to compress video, preserving the demanded high picture quality at acceptable computational cost.

Resumen

Los sistemas de TV digital modernos comprenden demodulación digital y requieren procesado de señal audiovisual incorporando una arquitectura de memoria unificada para todo el sistema. El cuello de botella de estos sistemas en chip (SoC) es la conexión con dicha memoria externa. Debido a que la señal de video consume la mayor parte del ancho de banda disponible, la compresión interna de esta señal permite la reducción del ancho de banda necesario, permitiendo sistemas donde el coste es aceptable. La compresión de video en sistemas embebidos debe ser visualmente sin pérdidas y de coste computacional aceptable. Esto requiere que la etapa de decorrelación de la señal de video sea extremadamente eficiente para limitar la cantidad de cuantización necesaria, evitando que la calidad de la imagen resultante llegue a niveles inaceptables. En este proyecto, un estudio de caracterización de la señal de video es llevada a cabo, resultando en nuevos conceptos en decorrelación de video. Basándose en estos conceptos, ha sido desarrollado, implementado y testeado un codec que incorpora predicción en el dominio espacial, DCT y que explota las características simétricas y la repetición de estructuras en la información de video. A partir de los resultados puede concluirse que las características de simetría a veces presente en los bloques de una imagen resultan ser una manera eficiente para comprimir video, preservando la calidad exigida a un coste computacional aceptable.

Contents

1 Introduction

The past twenty years, television went through an evolution that transformed a fully analog system into a fully digital system. Digitizing video led to the introduction of higher refresh rates, resulting in very complex and expensive televisions. Very Large Scale Integration (VLSI) fueled the integration process, whereas video compression techniques enabled a full digital transmission systems, resulting in System-on-Chip (SoC) solutions that accommodate channel decoding, source decoding and state-of-the-art television signal processing. Such a high integration of television functionality requires system solutions, in which all required memory is placed outside of the SoC. In such an unified-memory architecture, the video subsystems are the dominant factors in memory bandwidth consumption. The required memory bandwidth influences the width of the data bus and the number of pins on the SoC and thereby the package type of that SoC. Furthermore, the package type directly influences the number of layers for the Printed Circuit Board (PCB). All these technical aspects are influenced when applying Embedded Video Compression (EVC). The cost of EVC, compared to the total system cost that can be reduced, however, justifies this approach.

To battle the bandwidth in such an unified-memory based SoC, EVC is applied. Applying video compression inside an SoC is a trade-off between reducing video bandwidth on one hand and decreasing picture quality on the other hand. This becomes even more relevant for the situation in the nearby future where the number of bits per sample will be further increased resulting in 36 bits per sample. Taken into account the increase in frame rate from 50 Hz to 200 Hz, video traffic will be the dominant factor in an SoC. On the other hand video compression inside an SoC for HD television must be transparent for the user. It is for this reason, that video compression must be visual lossless, such that is does not degrade the perceived video quality. With EVC, the focus lays on extreme high video quality at modest compression ratio.

For a digital television SoC the video input may originate from a Digital Video Broadcast (DVB) or may enter the set via an external input from e.g. a PC or a video signal generator. The first video source has been compressed before, whereas video from an external input allows the full spectrum of video sequences, compressed or uncompressed, to enter the digital television. As a consequence, EVC has to deal with a broad range of video sources and the corresponding video characteristics.

A video sequence can be noisy or noise free, originate from a camera or be computer generated and can be natural video oriented or have a graphical characteristic. Visual lossless video compression is a challenging research topic, which requires a thorough understanding of the characteristics of a possible video sequence. This thesis deals with compression of high dynamic range video using 30 or 36 bits per video sample. Section 2 introduces some basic concepts about image and video compression later used in the thesis as decorrelation methods, quantization, etc. In Section 3 different ways of measuring the quality of the restored image after the compression process are discussed. Section 4 discusses various color domains as RGB,

YUV, YC_oC_g used by H.264 and another one which is based on Hue Saturation Value (HSV). Finally in Section 5 a complete description of the CODEC for EVC developed in this thesis is presented, as well as the tools utilized in the process. Finally, Section 6 presents some conclusions.

2 Introduction to Image Compression

This chapter provides a brief introduction to the main image compression tools used in this thesis, being conscious that many others exist as wavelets, fractals, etc that are out of scope in this thesis. An image compression scheme consists of three basic building blocks as depicted in Figure 1. The original image data will have a certain entropy H_0, which is reduced in each of the processing steps with a certain efficiency η. The purpose of the decorrelation block is to reduce the required bits per pixel and to a large extend determines the efficiency of the video coding chain. When the de-correlated output data is not quantized, video is losslessly coded. The compression ratio then depends strongly on the image content and therefore will be unpredictable, which limits broad usability. To have more control over the compression ratio, the de-correlated video data is quantized and the video signal is then lossy coded. The last step in the coding chain is the entropy coding block, which uses the statistical properties of the processed data to further reduce the bit cost.

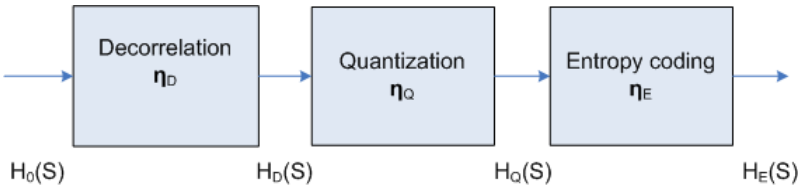


Figure 1: Basic image compression process chain

2.1 De-correlation

The de-correlation is usually the first step applied on image data. The purpose of this step is to eliminate correlation between pixels to enhance the efficiency of the entropy encoder. In natural video, neighboring pixels are often similar to each other. As a result, pixels contain a certain amount of redundancy. The objective of the de-correlation processing step is to reduce this amount of redundancy. The efficiency of the de-correlation step can be determined by calculating and comparing the entropy of the correlated and the de-correlated data. The entropy of a data set is calculated by Equation 1 [1] where p_i is the probability of occurrence of symbol i.

$$S = \sum_{i}^{N} p_i \frac{1}{log_2(p_i)} \tag{1}$$

The entropy calculated according by Equation 1 is a theoretical lower bound on the minimal required bits to represent a symbol. It is for this reason that the entropy is used to obtain a forecast regarding the achievable compression ratio. Moreover, it enables to determine the bit cost prior to entropy coding such as Huffman [2], Rice [3] or arithmetic coding [4].

A basic approach for de-correlation is to not transmit the absolute pixel values but the difference value between the current and the previous pixel. An example of this method is shown in Figure 2. Obviously for the first pixel value a difference can not be calculated so its absolute value is used instead. In a practical implementation the decoder must be notified on the applied de-correlation method, which introduces some overhead. In this thesis different de-correlation methods will be presented and examined. It is obvious that there will be different de-correlation methods for different image data. The topic of this thesis is to find suitable de-correlation methods for any image type and combine them.

2.1.1 Differential Pulse Code Modulation (DPCM) and prediction

The concept behind Differential Pulse Code Modulation (DPCM) [5] is to transmit a differential value for two pixels where the relation between these two pixels is dependent on the DPCM algorithm. Figure 2 depicts a basic DPCM encoder and decoder block diagram. For an DPCM encoder according to Figure 2 the first pixel

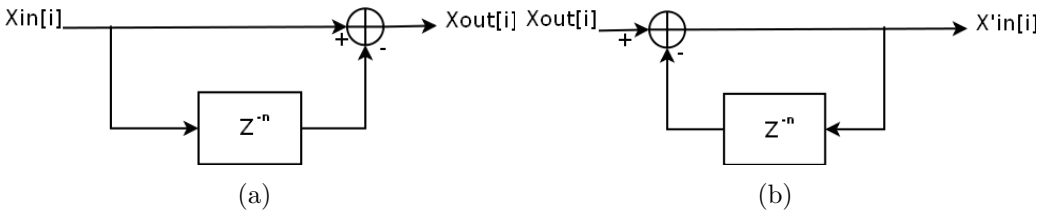


Figure 2: (a) Forward DPCM in the coder and (b) Backward DPCM in the decoder

is either transmitted as absolute value or as a difference value. For the latter case, the predictor can be a calculated reference value. An array x of pixels with the index i can be transformed into an array y of DPCM values and vice versa by using Equation 2 and Equation 3.

$$y[i] = x[i] - x[i-1] \tag{2}$$

$$\hat{x} = \hat{y} + \hat{y}[i-1] \tag{3}$$

A more sophisticated form of DPCM uses a variable step size n between the two pixels which are subtracted, as indicated in Figure 3. This method is very effective if the processed video data has repetitive structures e.g. checker-boards. When going

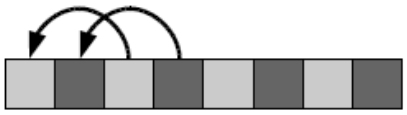
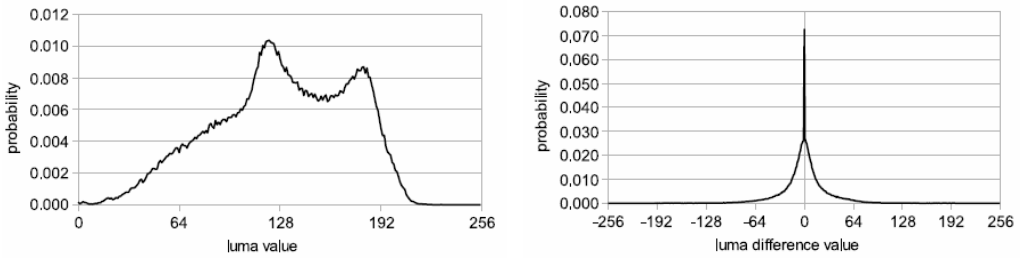


Figure 3: DPCM with adaptive step size (step size = 2)

to two dimensional blocks, the efficiency of DPCM de-correlation can be increased.

Now not only the step size for the subtraction can be optimized, but also the scan pattern can be tailored to match the image content. This allows multiple subtraction patterns in the spatial block region increasing the probability of matching the image content. Objective of DPCM, equally to all the other de-correlation methods, is to change the probability of the symbol occurrences, such that it matches the entropy coding step. It is due to the large similarity of adjacent pixels, in most images, that the difference values will be small or even zero. This packs the Probability Density Function (PDF) of the symbols (see Figure 4) such that the majority of values are concentrated around zero.



(a) (b)

Figure 4: Influences of DPCM coding on PDF. (a) PDF of PCM values of an image showing natural video content. (b) PDF of DPCM values of the same image

Moreover, prediction technique can be seen as a type of DPCM where the information to be subtracted to the original data has been some way preprocessed in order to exploit the existent correlation. The objective of prediction is to calculate a predictor that matches the actual segment content. As video consists of successive 2D images, that show a high correlation in spatial as well as temporal domain, there are three dimensions from which a predictor can be selected. One dimensional prediction exploits information only from the current line. Two dimensional prediction exploits information using multiple lines and three dimensional prediction uses information from previous or next image to predict the current image. Main objective of prediction is to create a predictor, that, when subtracted from the actual segment data (DPCM), results in a residual information signal with a substantially lower absolute difference value. H.264 offers spatial prediction on a 4×4 pixel block basis, as a processing step prior to the DCT transform. The spatial prediction techniques uses pixel information from previously coded blocks to form the predictor. Temporal prediction is outside the scope of this project, because the focus is on spatial compression due to the embedded nature of the codec, in which temporal prediction would require a greater amount of memory to store previous frames. In general the encoder incorporates a defined set of prediction modes and parameters. For the prediction, the encoder applies all available prediction modes and selects the best matching candidate on the basis of a metric.

2.1.2 Discrete Cosine Transformation (DCT)

The Discrete Cosine Transformation (DCT)[6] is a linear, orthogonal transform which expresses the image data as a series of cosine functions, representing the occurring spatial frequencies. It is related to the Discrete Fourier Transform(DFT) but has some advantages making it more suitable for image compression. The DCT is using real numbers for calculation and is about half the size of a comparable DFT. This makes it suitable for low-cost implementation. There are eight standard variants of the DCT where usually two are used for image processing. The DCT-II is used for transforming the image data into the frequency domain.

$$X_k = \sum_{n=0}^{N-1} x_n \cos[(2n+1)k\frac{\pi}{2N}] \mid k = 0, ..., N-1 \tag{4}$$

The DCT-III is also called inverse DCT and is used to transform the data back into the spatial domain.

$$X_k = \sum_{n=0}^{N-1} x_n \cos[(2k+1)n\frac{\pi}{2N}] \mid k = 0, ..., N-1 \tag{5}$$

To enable usage in block-based image compression, it is beneficial to extend the one-dimensional DCT in Equation 4 to a two-dimensional DCT, calculated by Equation 6.

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos[(2k_1+1)n_1\frac{\pi}{2N_1}] \cos[(2k_2+1)n_2\frac{\pi}{2N_2}] \tag{6}$$

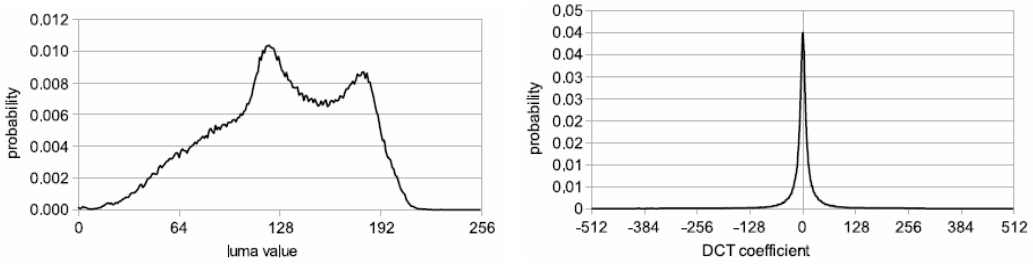
The DCT may be calculated according to Equation 4 or by applying a matrix-based operation. To perform the two dimensional DCT, a two-step matrix multiplication in the form $Y = CXC^T$ is used. The inverse operation becomes then $X' = C^TYC$. Hereby X represents the input data in spatial domain, Y represents the DCT output, C corresponds to the coefficient matrix and C^T is the transposed version of C. This requires that matrix C is orthogonal which is not the case for Equation 4. Orthogonality of Equation 4 is achieved by multiplying scaling factors of $\sqrt{\frac{1}{N}}$ to the X_0 term and $\sqrt{\frac{2}{N}}$ to all the remaining terms. The coefficients of matrix C are then calculated to

$$\begin{cases} c_{i,j} = \sqrt{\frac{1}{N}}, & \text{if } \; i = 0, 1, ..., N-1; j = 0; \\ c_{i,j} = \sqrt{\frac{2}{N}} \cos[(2i+1)j\frac{\pi}{2N}], & \text{if } \; i, j = 0, 1, ..., N-1; \end{cases} \tag{7}$$

where $c_{i,j}$ defines the matrix element in column i and row j in a matrix of the size $N \times N$. An example of a 4×4 matrix is depicted in Equation 8

$$C = \begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}, \quad \begin{matrix} a = 1/2 \\ b = \sqrt{1/2}\cos(\pi/8) \\ c = \sqrt{1/2}\cos(3\pi/8) \end{matrix} \tag{8}$$

From Equation 8 it becomes apparent, that the calculation requires floating point operations. It is for this reason, that H.264 introduced a modified DCT calculation, based entirely on fixed-point arithmetics. This is achieved by factorizing the matrix multiplications and combining the cosine terms with the quantization step in an integer look-up table. For the matrix multiplication itself, only additions and shift operations are required. This topic will be more deeply explained in a future section. The purpose of transforming the image data into the frequency domain is fueled by the fact that the Human Visual System (HVS) is less sensitive for high detail in natural video, which corresponds to the high frequencies in the transform domain. Decomposition of a region in the spatial domain using a 2D-DCT results in separation of the high and low detail. Moreover, the low detail is clustered in the upper-left corner of the DCT block and the high frequency is distributed over the remaining part of the DCT block. This frequency decomposition allows tailored quantization of each individual frequency component according to the sensitivity of the human eye (HVS).Although a DCT transform expands the required number of bits to represent the output symbols, the entropy in natural video is often reduced even without applying quantization, indicating the good decorrelation performance of the DCT. This is due to the concentration of the block energy in the lower frequencies and thus modifying the PDF as shown in Figure 5.



(a) (b)

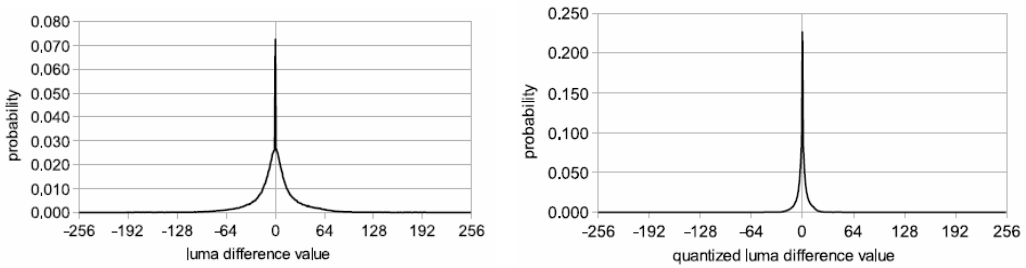
Figure 5: Influences of DCT on PDF. (a) PDF of PCM values for an image containing natural video content. (b) PDF of DCT coefficients for the same image. (Note, that the luma values have been biased by subtracting 128 prior to transformation to create balanced DCT output.)

However, high spatial frequencies not only occur in regions, where pixel values are changing rapidly, but are also present in sharp edges. Applying quantization in the DCT domain, to reduce the amplitudes of high spatial frequencies, distorts sharp edges creating visible artifacts around the borders.

2.2 Quantization

De-correlation in general is a lossless data transformation. Except for some rounding errors in the transformation calculations, the image can be reconstructed from the de-correlated data set without introducing distortion. Depending on the image

content, the compression ratio widely fluctuates within certain boundaries. While some graphical images can be compressed successfully using only de-correlation and entropy coding, most of the images with natural content can not be compressed sufficiently on the basis of only these methods. If redundancy reduction, which is the purpose of the de-correlation step, is not sufficient, the next step in the coding process is quantization, which removes irrelevance and possibly also relevant information. While redundancy can be described on a mathematical basis, this approach is not feasible for irrelevance. This is caused by the fact, that irrelevance is subjective by nature and depends on the Human Visual System (HVS). Visual lossless video compression is achieved when only irrelevant information of the image data is removed. The challenge in visual lossless video compression is the development of a coding scheme resulting in a pre-determined compression ratio for a broad range of video sources. The difficulty in defining a coding scheme as visually lossless is, that different people perceive images in a different way. Distortions, not visible for some people might be visible for others. In such cases, studies with many different people have to be conducted to obtain accurate results. If that is not possible the threshold between visually lossless and lossy needs to be estimated. Basically, quantization is the reduction of the accuracy of a data set by discretization of the values a signal can take. In image compression, this data set consists e.g. of the original pixel values or a transformed version e.g. DCT coefficients of these pixels. In the quantizator [7], the dynamic range of the data set is divided by a quantization step and all the values of the image contain in a certain quantization step are set to the correspondence value. Accuracy reduction introduces artifacts (noise), which degrades the quality of the reconstructed image. The main objective of quantization is the modification of the image data set, to positively influence the PDF. As depicted in Figure 6 where the modification of the PDF of the DPCM values depicted in Figure 4 is shown, this results in concentrating the symbol distribution, regardless of the applied de-correlation, although the de-correlation, to a large extend, determines the efficiency of the complete coding scheme. The specific quantization method used in this thesis [17] will be explain in following sections.



(a) (b)

Figure 6: Influences of quantization on PDF. (a) PDF of un-quantized DPCM values of an image showing natural video content. (b) PDF of quantized DPCM values of the same image

3 Image Quality Assessment

In this chapter a brief overview on how to examine the quality of compressed images is given. When using lossless compression techniques this step is not necessary since the image, which is reconstructed from the compressed data, is equal to the original one. However, this does not apply to visually lossless image compression. The difference between those two types is, that in the latter case there is no visual difference between reconstructed and the original image but there may be an actual difference. It is almost impossible to judge image quality solely by examining numbers. Therefore in this thesis the image quality is assessed by objective and subjective methods, described in the following sections.

3.1 Objective Quality Assessment

Objective image quality assessment is not a trivial task. It is not sufficient to just calculate the differences between reconstructed and original image to determine the quality perceived by the human eye. The problem with these techniques is, that the human eye perceives a certain deviation differently depending on where the region of interest (ROI) is located in the image. However, techniques such as e.g. Peak Signal to Noise Ratio (PSNR) are still widely used metrics to accomplish image quality assessment. The metrics used in this thesis are described in subsubsection 3.1.1 and 3.1.2. Image quality can also be described by methods other than differences between reconstructed and original image. In the field of image compression an other way of defining image quality is to examine artifacts introduced by the compression algorithm. In some cases this can even be done without having a reference image but it is certainly more precise when one is used. The most common artifact in block based image compression schemes is the introduction of blocking artifacts. These can be seen when neighboring blocks have large differences in their average luminance so that one can see the boundaries of certain blocks. A method on how to define the output image quality based on blocking artifacts is described in subsubsection 3.1.3.

3.1.1 Measuring Image Differences

The most straightforward way to examine the quality of a processed image is to compare it to a reference image. There are, though, different approaches on how the differences are weighted in respect to the image quality. The most common measure of image quality is the Peak Signal to Noise Ratio (PSNR). It represents the ratio between the maximum occurring luminance value and the magnitude of the luminance differences, determined by the Mean Square Error (MSE). For an image with the dimensions $N \times M$ the MSE can be calculated with Equation 9.

$$MSE = \frac{1}{NM} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |x_{i,j} - \hat{x}_{i,j}|^2 \tag{9}$$

The PSNR can then be calculated with Equation 10 where max refers to maximum luminance value in the image, usually $2^{\text{bit depht}}$.

$$PSNR = 10log_{10}\left(\frac{max^2}{MSE}\right) \tag{10}$$

3.1.2 Difference Bookmarks

Following the same line that in the previous section, there are three bookmarks used in this project to measure the amount of changes produced in the restored image in comparisson with the reference. These bookmarks are:

- **Maximum Difference**: it corresponses with the maximum pixel difference within the whole image.
- **Average Difference**: this is the average difference considering all the pixels differences in the image.
- **Amount of changes**: it is the amount of changed pixes, i.e. not exactly restored pixels, in the whole image.

3.1.3 Measuring Blocking Artifacts

A common problem with single-block-based compression schemes is the introduction of blocking artifacts when using lossy compression as can be seen in Figure 7. The reason for this is the independent quantization of each block. Blocking artifacts occur first in rather flat blocks where the high frequency part of the DCT transformed image block is low. With an increasing quantization factor, these blocks tend to be rendered uniform with a luminance near their DC value. Since the DC value of two neighboring blocks can be different, the result will be two almost uniform neighboring blocks with differences in luminance and a visible luminance difference between them, which is noticeable by the human eye. To measure the blockiness of an image, the blockiness of each individual block is determined and used to calculate the overall blocking index of the image.

3.2 Subjective Quality Assessment

As it is pointed out in subsection 3.1 objective evaluation of image quality is almost impossible or involves high calculation effort. The objective image quality assessment methods are only needed, if a lot of images need to be examined in a short time. The most accurate method to assess image quality is still the examination by human observers. If the amount of images to examine is low, this is still a reasonable way to perform the task of image quality assessment. Furthermore it can be used to verify the results of the objective methods. In this thesis most of the initial quality

Figure 7: (a) Original image and (b) block artifacts due to independet quantization in restored image

examination for new image processing techniques is done on a subjective basis by comparing a processed image with a reference. However, subjective examination still has the disadvantage that different people judge the results very differently. To limit this effect there are a few techniques, which can be used to achieve more accurate results. A very common problem with judging image quality, when the images only have slight differences, is to find the regions where visible artifacts are occurring. From common sense it is clear, that it is easy to see certain artifacts when one knows where to look for them but they are not that obvious if one does not know where they are. To help eliminating this problem, the differences and artifacts need to be visualized so that it can be easily seen, where the strongest differences are located. One possibility to do that is using a so-called difference image (Figure 8). It shows only the absolute values of the differences between the original and the reference image. For images with only small differences the difference values can also be amplified to help distinguishing between error levels. Such an image shows the regions where to look for artifacts in the processed image so the specific regions can be magnified for a closer inspection. An other method of finding regions with artifacts is the process visualization. That means that areas, processed by a certain algorithm, are highlighted, whereas untouched regions stay black. Obviously, this only works for processing schemes using different algorithms for different regions.

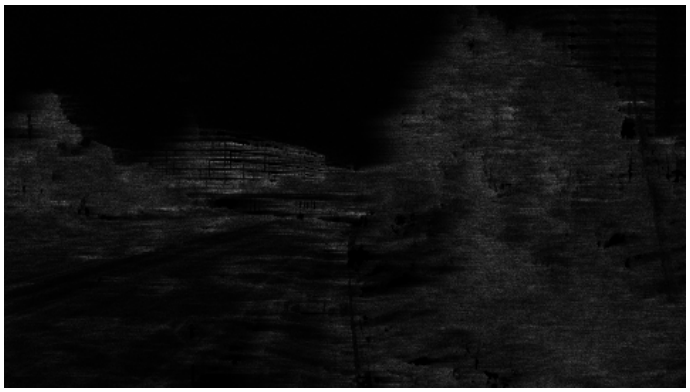


Figure 8: Difference image between original and recovered images

4 Color spaces

A color space is a model in which the human perception of color is described using a set of mathematical equations. In practice a minimum set of three equations is required to describe the whole color space. To understand a color space knowledge of the Human Visual System (HVS) [8] is required. The human eye is constructed of photo sensible cells called cones, whereby the name reveals the shape of the cell, that are responsible for the color perception. These cells are sensitive to a specific wavelength. There are three kind of cones, each operating as a bandpass filter. As a result, each cone is sensitive to a specific wavelength. The S (short-wavelength) cones are sensitive around the 420 nm, which corresponds to the violet light. The L (Long-wavelength) cones are sensitive around 564 nm, which corresponds to the yelowish-green light. Finally, the M (middle-wavelenghts) cones are sensitive around 534nm, which corresponds to the green light.

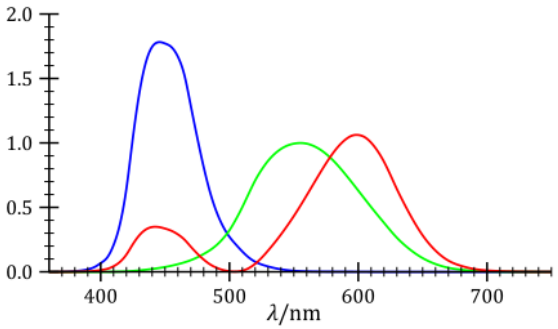


Figure 9: Cones Absortion

This way, any kind of light perceived by the eye is decomposed in just three signals generated by each cone, called tristimulus. The combination and processing of these signals by the brain result in what is called color sensation.

Other photosensitive cells called rods are present in the eye, but they have a different function. They act in a different band, near 500 nm, and are much more sensitive to the light, even reaching saturation in many cases. Although these cells form a group they do not have sufficient capacity to distinguish little details. Rods operate in poor light conditions and do not detect colors.

The fact that only three information signals describe the visual spectrum, allow the setup of an Euclidean space using the stimuli of each of the cones as x,y, and z axes. The origin of this space is described by (0,0,0) and corresponds to the black color, colors can be represented as three coordinates or vector. These vectors have the property of being additive, enabling the description of any visual color perceived by the HVS. When restricting the tristimulus range a specific color gamut is created.

The human color space is a horse-shoe-shaped cone extending from the origin to, in principle, infinity as depicted in Figure 10. In practice, the human color receptors will be saturated or even be damaged at extremely-high light intensities, but such

behavior is not part of the color space and neither is the changing color perception at low light levels. Practical color spaces represent a portion of this space.

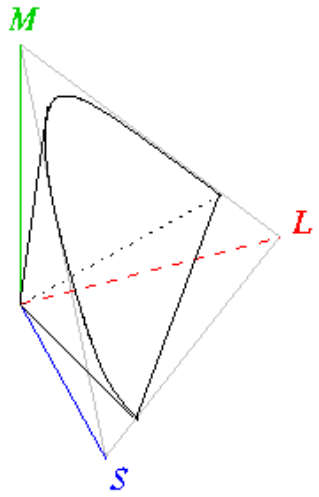


Figure 10: Human Color Space

In order to find a color space where the combination of image representation and the performance of different compression tools achieve a better compression rate mantaining an acceptable image quality, different color spaces will be investigated and tested. In the following sections four different color spaces are discussed.

4.1 *RGB* Color space

Each component of the RGB color space represents one of the signals produced by the cone cells in the human eye. RGB, which stands for red, green and blue, is usually the original color space for image and video data. On one hand this is due to the extended use of Bayer filter mosaics in the manufacturing of the Charge Coupled Devices (CCD) frequently used in cameras. A Bayer filter mosaic is a Color Filter Array (CFA) for arranging RGB color filters on a square grid of photosensors. Moreover, the RGB color space is also dominantly used in graphics for Internet applications.

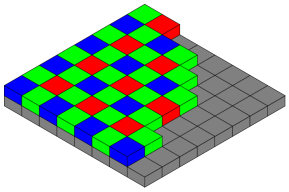


Figure 11: Bayer Pattern

The RGB color space shape is a cube, as each signal forms an axes orthonormal to the others two. In this color space there is no distinction between luminance and chrominance components, although G signal contains the most of the luminance information in comparison with the R and B signals.

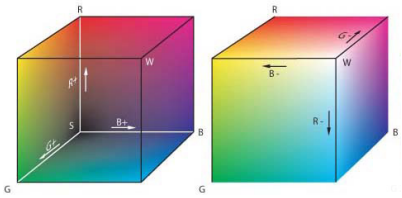


Figure 12: RGB color space

In order to measure the characteristics of each color space, a set of test images listed in Table 1 is used and analyzed using histogram and entropy method. The entropy term by itself in the information theory context usually refers to the Shannon entropy, which quantifies, in the sense of an expected value, the information contained in a message, usually in units such as bits. The larger its value is, more unexpected the information is. The entropy expression in Equation 1 when applied to an image can be written as a Equation 11:

$$S = \sum_{i=0}^{H*W} p_i \cdot log_2 \frac{1}{p_i} \tag{11}$$

$$p_i = \frac{\#\ of\ occurrences\ of\ pixel\ i}{H * W} \tag{12}$$

where p is the probability of a certain pixel within the test image, W is the width of the image and H its height, both in pixels.

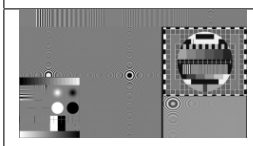
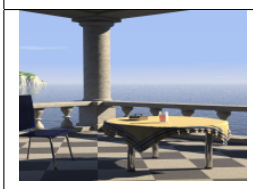
Image	Name	Bit Depth	Caractheristics	S(R)	S(G)	S(B)
	freeway	10	Natural video	9.4553	9.5176	9.2293
	plants	12	Natural, computer originated	10.7396	10.6875	10.9820
	boursorma	10	Graphics, text, natural	6.7279	6.9725	6.7607
	testpic	10	Graphic, artificial, just luminance	7.5095	7.5095	7.5095
	balcony	12	Natural, computer originated	10.1197	9.9019	9.7791

Table 1: Test images set and Channel Entropy

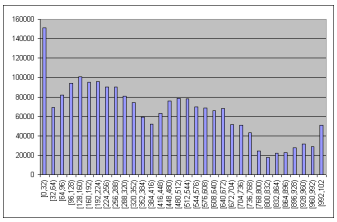
The histogram of an image attempts to show the Probability Density Function (PDF) of the pixels on the image. Pixel values are organized in categories. Each bar corresponding to a bin of a certain width indicating the amount of occurrences of a certain value or value(s) in the picture. The number of bins has been chosen following Equation 13, and the width of each bin is determined by Equation 14.

$$B = \#bins = \left\lceil \sqrt{\#\,possible\ values} = \sqrt{2^{Bit\ Depth}} \right\rceil . \tag{13}$$

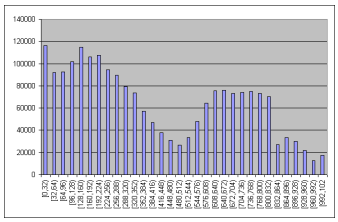
$$W_b = \left\lceil \frac{\#\,possible\ values}{B} \right\rceil . \tag{14}$$

The histograms of the correspondence test image in RGB color space are shown in Figure 13. Note that the Testpic image has only one histogram in Subfigure 13(m),

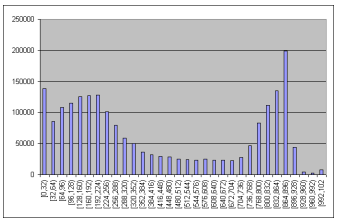
since it is a gray scale image, which results that the three components RGB are equal.



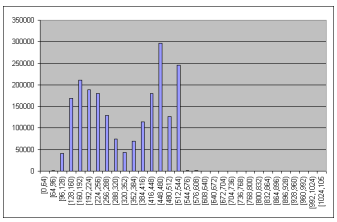
(a) Freeway Channel R



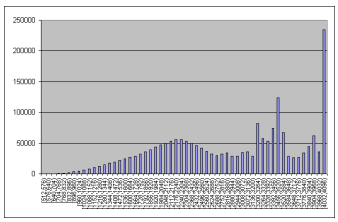
(b) Freeway Channel G



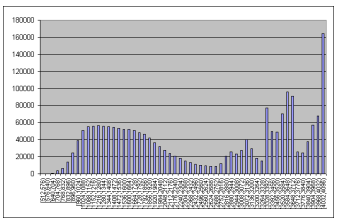
(c) Freeway Channel B



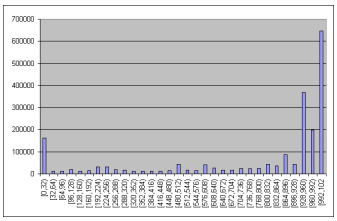
(d) Plants Channel R



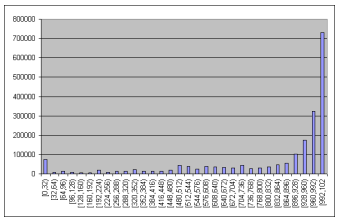
(e) Plants Channel G



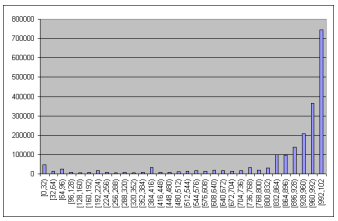
(f) Plants Channel B



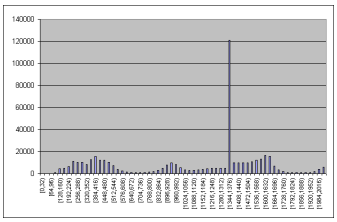
(g) Boursorma Channel R



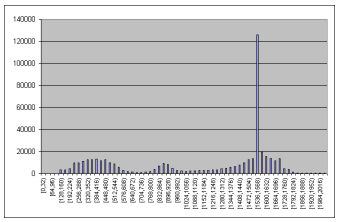
(h) Boursorma Channel G



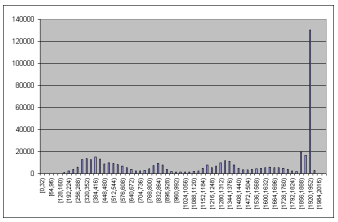
(i) Boursorma Channel B



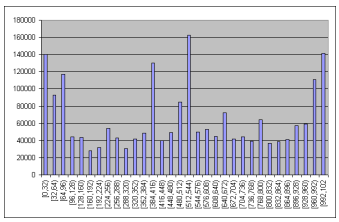
(j) Balcony Channel R



(k) Balcony Channel G



(l) Balcony Channel B



(m) Testpic Channel R,G,B

Figure 13: Histograms of the test set in RGB

4.1.1 Inter plane prediction in RGB Color space

RGB has been always considered as a bad color domain to achieve compression since its three components are high correlated and so, there is a lot of redundancy information. That is exactly what the prediction model that will be explained here[12], [14] tries to take advantage of. On one hand, it uses spatial correlation in an RGB plane to calculate the parameters it needs. On the other hand, the correlation within different planes is used to predict one plane from the other.

In this model, plane G will be use as base plane since the human eye is more sensible to it due to it contains the most of luminance information. In order to analyze the inter-plane correlation, correlation coefficients within G/R, G/B and B/R are calculated for each block using Equation 15.

$$\rho_{g/r} = \frac{M^2 \sum G_i R_i - \sum G_i \sum R_i}{\sqrt{(M^2 \sum G_i^2 - (\sum G_i)^2)(M^2 \sum R_i^2 - (\sum R_i)^2)}} \tag{15}$$

Where M is the block size and G and R indicate green and red planes respectively. The same equation can be used to calculate $\rho_{g/b}$ and $\rho_{b/r}$. In the calculation process of the correlation coefficient could be found that all the samples of a block are the same. If this is the case, it is possible to indicate it saving just one sample and then copying it in the restore step. In Table 2 the mean correlation coefficient per block is shown, not counting with this special cases.

Correlation Coefficient			
Image	G\R	G\B	R\B
balcony	0.9853	0.9352	0.9074
plants	0.9630	0.9123	0.9110
freeway	0.6599	0.6574	0.7250

Table 2: Intra-Plane Correlation coefficients.

The prediction model follows a linear equation and ω and o are calculated to minimize the Minimum Square Error (MSE) within the actual block and the predicted one, as shown in the following equations:

$$\widehat{r_i} = \omega g_i + o \tag{16}$$

$$min\ e_{\omega,o} = \sum_{i=0}^{N} (r_i - \omega g_i - o)^2 \tag{17}$$

To accomplish this, we find:

$$\omega = \frac{N \sum_{i=0}^{N} (g_i r_i) - \sum_{i=0}^{N} g_i \sum_{i=0}^{N} r_i}{N \sum_{i=0}^{N} g_i^2 - (\sum_{i=0}^{N} g_i)^2} \tag{18}$$

$$o = \frac{1}{N} (\sum_{i=0}^{N} g_i - \omega \sum_{i=0}^{N} r_i) \tag{19}$$

where g_i and r_i represents the reconstructed samples of the G plane and R plane and N is the number of samples of the block that are used to the calculation (the maximum value that it can take is M^2, i.e, the amount of pixels in a block). In this experiment N is 8 while M is 4 . g_i and r_i can be changed to b_i to calculate the parameters for the correspondence plane. There are 4 modes to calculate the weighting parameters that can be seen in Figure 14.

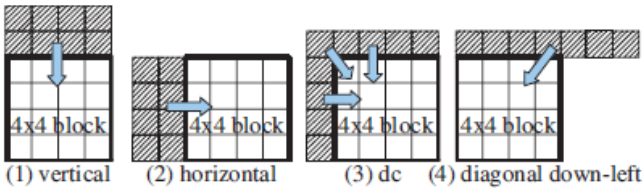


Figure 14: Four modes to calculate ω and o parameters

Since the four modes are used, it would be needed 2 bits per block to indicate which mode is used, in this case only horizontal and vertical mode will be used depending just on the position of the block in the image. This way if it is the first block of a line vertical mode is used, otherwise, the horizontal one is used.

In this point the parameters values should be transmitted since the current blocks are used to calculate them. To resolve this, taking advantage of spatial correlation, instead of calculating both parameters from the actual blocks, it will be done from the previous ones. This is, to calculate $\omega^i_{g/r}$ and $o^i_{g/r}$, the N pixels of r_{i-1} are used, where i means the index of the block inside the image.

After this process, after calculating the weighting parameters, the error of the prediction is calculated and transmitted. As the prediction is supposed to work well and be optimized to minimize this error, the values of this differences should be much smaller and have a lower dynamic range, reducing the entropy of the plane and the total image. On the other hand, G plane has not be processed in any way. So, here intra-prediction, conversion to another color space, DPCM or any other tool can be used to decorrelate such plane.

4.2 YC_bC_r Color space

The YC_bC_r color space is a widely deployed in video compression. It is the digital version of the YUV color space used for analog television, whereby Y is equal for both color spaces and C_b corresponds to U and C_r corresponds to V. The color space is composed of three components, a single Luminance channel and two chrominance channels, blue and red. Since HVS is much more sensitive to variations in Luminance,it is custom to subsample the chrominance components UV, resulting in the different extended formats 4:4:4, 4:2:2, 4:2:1, etc, where the value of numbers is the relation between the sample frequency in each plane Y,U,V. I.e., in 4:2:2, U and V components are sampled at half frequency compared to the Y component. This sub-sampling results in a compression gain but also a decreasing in the quality of the restored image since it requires prediction of the lost samples, which is extremely noticeable in computer graphics.

The forward transform to go from an RGB triplet to a YC_bC_r triplet is:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{20}$$

And the inverse transform is:

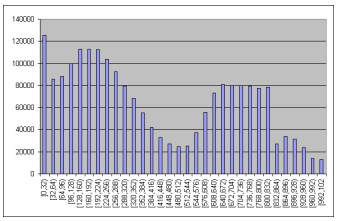
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.402 \\ 1 & -0.344 & -0.714 \\ 1 & 1.772 & 0 \end{bmatrix} \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} \tag{21}$$

Due to the transform coefficients are float, it is impossible to recover original image since a rounding error appears when pixel values are passed to integer.

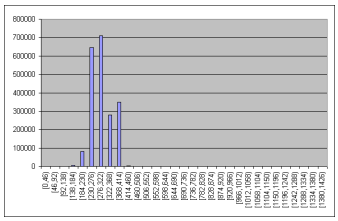
In table 3 the entropy values of the different images are depicted, and Figure 15 shows the correspondent histograms.

Entropy						
Image	R	G	B	Y	U	V
freeway	9.4553	9.5176	9.2293	9.7708	7.4601	6.9520
plants	10.7396	10.6875	10.9820	11.0016	8.9937	7.9414
boursorma	6.7279	6.9725	6.7607	5.7197	5.2861	5.1210
testpic	7.5095	7.5095	7.5095	7.5095	0	0
balcony	10.1197	9.9019	9.7791	9.9098	8.3060	7.6862

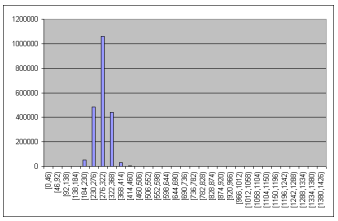
Table 3: Channel and Average Entropies of images set in RGB and YUV



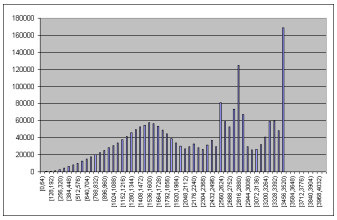
(a) Freeway Channel Y



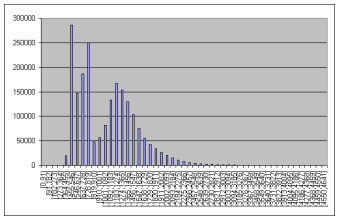
(b) Freeway Channel U



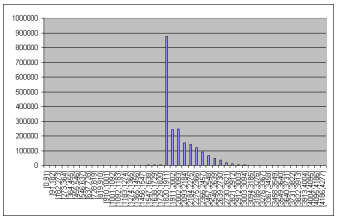
(c) Freeway Channel V



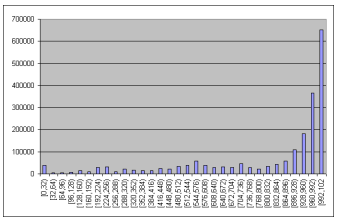
(d) Plants Channel Y



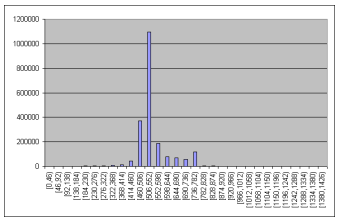
(e) Plants Channel U



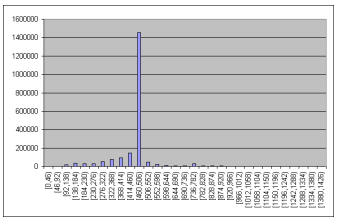
(f) Plants Channel V



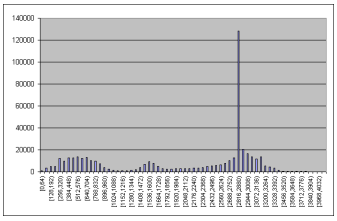
(g) Boursorma Channel Y



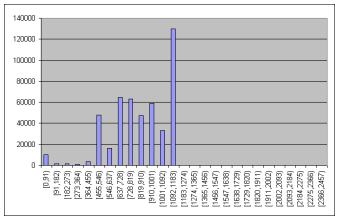
(h) Boursorma Channel U



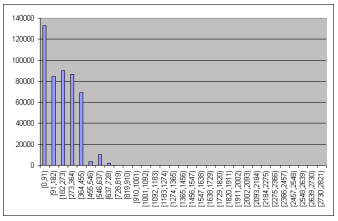
(i) Boursorma Channel V



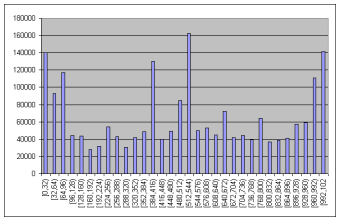
(j) Balcony Channel Y



(k) Balcony Channel U



(l) Balcony Channel V



(m) Testpic Channel Y

Figure 15: Histograms of the test set in YUV

4.3 YC_oC_g Color space

The color space subjected to research use three information signals to represent a pixel value. The bit depth of each information signal may vary all depending on the deployed calculation. The objective of a video codec is to reduce the amount of information. The first step in such a coding scheme is decorrelation. Decorrelation can be achieved in various stages of the video chain. The stage where decorrelation can be achieved is in the color space. The objective is define a color space where information signals show a better decorrelation. This section discusses the YC_oC_g color space [9],[10]. In a similar way as YC_bC_r, in YC_oC_g the original RGB data is transformed into a luminance component and two chrominance components. The latter two components correspond to orange and green. A RGB triplet is transformed in to the YC_oC_g triplet using the forward transform

$$\begin{bmatrix} Y \\ C_o \\ C_g \end{bmatrix} = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 0 & -1/2 \\ -1/4 & 1/2 & -1/4 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \tag{22}$$

The inverse transform is defined by

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{bmatrix} \begin{bmatrix} Y \\ C_o \\ C_g \end{bmatrix}. \tag{23}$$

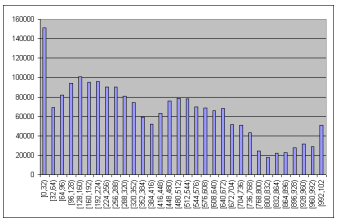
An efficient implementation of the RGB to YC_oC_g color space transformation requires only two shifts and four additions resulting in the following equations:

$$\begin{array}{rcl} C_o & = & R - B \\ t & = & B + (C_o >> 1) \\ C_g & = & G - t \\ Y & = & t + (C_g >> 1) \end{array} \quad \Leftrightarrow \quad \begin{array}{rcl} t & = & Y - (C_g >> 1) \\ G & = & C_g + t \\ B & = & t - (C_o >> 1) \\ R & = & B + C_o \end{array} \tag{24}$$

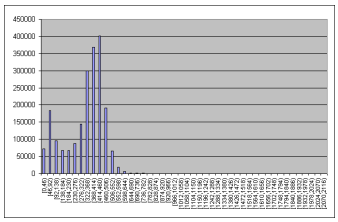
The dynamic range for the chrominance components in the new color space barely change. If the original image or video has N bits dynamic range for each RGB channel, then Y channel uses also N bits, but the Co and Cg channel requires an additional bit caused by the shift operation resulting in $N+1$ bits per channel. An important advantage of the YC_oC_g color space is that, although there is a 1 bit expansion in the dynamic range of the chrominance components, it is possible to recover the original RGB values since the transformation uses integer arithmetic, and so achieve better qualities in the final restored image.
Like in previous sections the color space is characterized using a histogram per test set image as depicted in Figure 16. For the calculation of this histograms the value $2^{Bit\ Depth}$ has been added to the pixels of the chrominance components in order to avoid negative values. Moreover, some bins at the beginning or at the end have not been displayed in the histogram since they were empty. Because the Testpic image is a black and white picture, only the histogram of the Y signal is shown since the chrominance components would be a Dirac delta in 0.

Entropy									
Image	R	G	B	Y	U	V	Y	C_o	C_g
freeway	9.4553	9.5176	9.2293	9.7708	7.4601	6.9520	9.7494	8.7293	7.2334
plants	10.7396	10.6875	10.9820	11.0016	8.9937	7.9414	11.0068	8.9204	8.3170
boursorma	6.7279	6.9725	6.7607	5.7197	5.2861	5.1210	7.0659	6.3810	5.6667
testpic	7.5095	7.5095	7.5095	7.5095	0	0	7.5095	0	0
balcony	10.1197	9.9019	9.7791	9.9098	8.3060	7.6862	9.8016	9.2566	7.3634

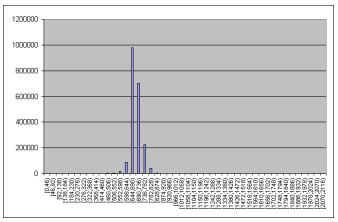
Table 4: Channel and Average Entropies of images set in RGB, YUV and YC_oC_g



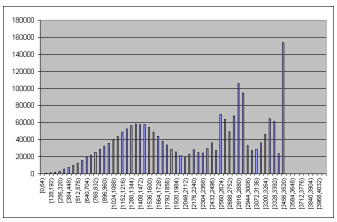
(a) Freeway Channel Y



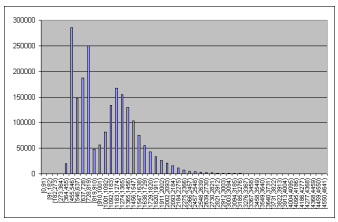
(b) Freeway Channel C_o



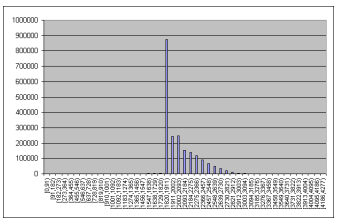
(c) Freeway Channel C_g



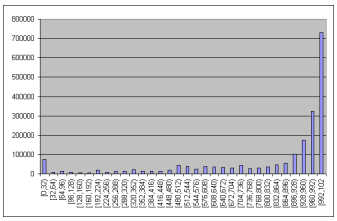
(d) Plants Channel Y



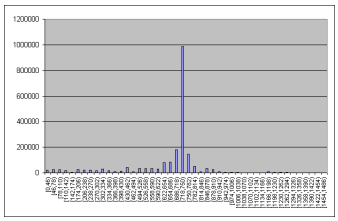
(e) Plants Channel C_o



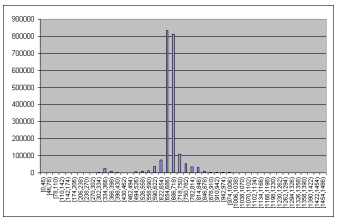
(f) Plants Channel C_g



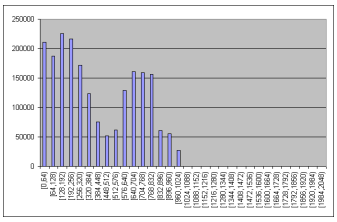
(g) Boursorma Channel Y



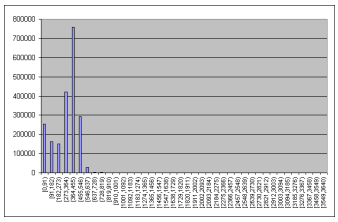
(h) Boursorma Channel C_o



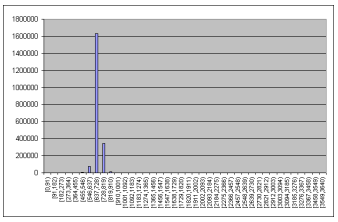
(i) Boursorma Channel C_g



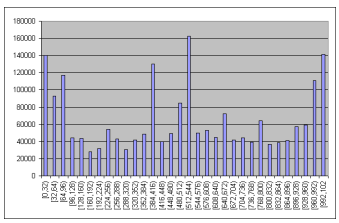
(j) Balcony Channel Y



(k) Balcony Channel C_o



(l) Balcony Channel C_g



(m) Testpic Channel Y

Figure 16: Histograms of the test set in YC_oC_g

4.4 HSV Color space

HSV color space is the abbreviation for the Hue, Saturation and Value based color space. This space is part to a category called perceptual color spaces, because it shows a high resemblance with the HVS, which also use hue, saturation and an intensity value. RGB depicts any color at three color primaries red, green and blue. HSV describes a color in terms of hue, which is exactly the color, the amount of saturation, which is the amount of white in this color, and the value that indicates the brightness of the color.

The transform for going from RGB to HSV and vice versa is not linear, since these two color spaces describe colors in a different way. The forward RGB to HSV transform is:

$$RGB_{max} = max(R, G, B) \tag{25}$$

$$RGB_{min} = min(R, G, B) \tag{26}$$

$$V = RGB_{max} \tag{27}$$

$$S = \begin{cases} 1 - \frac{RGB_{min}}{RGB_{max}}, & \text{if } RGB_{max} > 0; \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

$$H = \begin{cases} 0, & \text{if } RGB_{max} = RGB_{min}; \\ (60° \times \frac{G-B}{RGB_{max}-RGB_{min}} + 360°) mod\, 360°, & \text{if } RGB_{max} = R; \\ 60° \times \frac{B-R}{RGB_{max}-RGB_{min}} + 120°, & \text{if } RGB_{max} = G; \\ 60° \times \frac{R-G}{RGB_{max}-RGB_{min}} + 240°, & \text{if } RGB_{max} = B; \end{cases} \tag{29}$$

This space has a cylindrical shape [11]. The resulting circle of an horizontal cut perpendicular to the axis, contains all the colors of the space. Each one is in a different hue angle from 0° to 359°. In the radial direction there is the saturation, going from the center of the circle to the border, saturation goes from 0 to 1, so from white to the most saturated color in the specific hue angle. Finally, in the vertical direction, there is the value, which describes the brightness of the color.

The big advantage of this space is that its components are high decorrelated since each one of them represents a very different characteristic of a color. In addition, because of its own definition the dynamic range needed to code the whole space is much smaller than the original RGB for high-dynamic range video, i.e 10 to 12 bits per RGB component. This is on one hand side caused by the Hue component, which is an angle in degrees, so its dynamic range is [0,359], requiring only 9 bits for coding. On the other hand, Saturation is defined in the range [0,1], which in these experiments is coded with 8 bits. Finally, the Value parameter has the same dimension as the original RGB signal, since it is the maximum value in each RGB triplet. So, with this configuration, going from RGB with 12bps to HSV would mean going from $3 \times 12 = 36$ to $9 + 8 + 12 = 29$ bits. All these decorrelation and reduce dynamic range properties are reflected in the entropies of the images. The next tables shows the comparison between the entropy values of test images in the original RGB, YUV, YC_oC_g and HSV spaces.

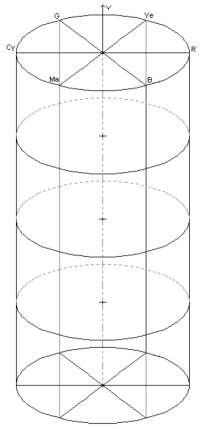


Figure 17: Cylindrical shape of HSV space

Entropy												
Image	R	G	B	Y	U	V	Y	C_o	C_g	H	S	V
freeway	9.4553	9.5176	9.2293	9.7708	7.4601	6.9520	9.7494	8.7293	7.2334	7.5291	7.1954	9.4433
plants	10.7396	10.6875	10.9820	11.0016	8.9937	7.9414	11.0068	8.9204	8.3170	5.6263	6.6797	10.5995
boursorma	6.7279	6.9725	6.7607	5.7197	5.2861	5.1210	7.0659	6.3810	5.6667	5.0560	5.2399	6.5952
testpic	7.5095	7.5095	7.5095	7.5095	0	0	7.5095	0	0	0	0	7.5095
balcony	10.1197	9.9019	9.7791	9.9098	8.3060	7.6862	9.8016	9.2566	7.3634	4.1488	5.8330	9.7013

Table 5: Channel and Average Entropies of images set in RGB, YUV, YC_oC_g and HSV

A popular way to compress a video signal is by dividing the spatial region in blocks, which are then subjected to further individual processing. When using block-based compression, the HSV color space may be opportune. This is based on the observation that components are much more static, i.e., it is not probable that saturation goes from almost 0 to unity, etc. When working on 4×4 blocks, there is a low deviation with respect to the mean value. This is true for the HSV components Hue, Value and Saturation, where the amount of deviation increases in this order. This can be usable to decorrelate even more the image, for example taking advantage of DPCM tools or DCT, since only the DC and very low frequencies will have non zero coefficients. This behavior could be opportune when using a transform kernel such as Discrete Cosine Transform (DCT). However disadvantage of such a transform is that it is lossy. Therefore, it is not possible to recover the original exact image if integer arithmetic is used. The limited quantity of bits to code the Saturation component and the fact of using division operations involve that there will be errors in the recovered image. The same problems appear in the inverse transform that is shown in next line:

$$h = \left\lfloor \frac{H}{60} \right\rfloor mod\, 6 \tag{30}$$

$$f = \frac{H}{60} - h \tag{31}$$

$$p = V(1 - S) \tag{32}$$

$$q = V(1 - fS) \tag{33}$$

$$t = V(1 - (1 - f)S) \tag{34}$$

$$h = \begin{cases} & R = V \\ 0, & G = t \\ & B = p \\ \\ & R = q \\ 1, & G = V \\ & B = p \\ \\ & R = p \\ 2, & G = V \\ & B = t \\ \\ & R = p \\ 3, & G = q \\ & B = V \\ \\ & R = t \\ 4, & G = p \\ & B = V \\ \\ & R = V \\ 5, & G = p \\ & B = q \end{cases} \tag{35}$$

(36)

Although all these drawbacks, with this configuration (9 bits for H, 8 for S and 12 for V for a 12 bit RGB image), the PSNR obtained is good enough not to visually appreciate errors in the recovered image. In the following table 6, results of applying forward and inverse transform are depicted. Values arrive almost to 60dBs.

PSNR (dBs)				
Image	R'	G'	B'	$R'G'B'$
freeway	61.67	56.19	62.71	60.19
plants	55.87	59.77	59.75	58.46
boursorma	60.37	51.87	59.43	57.22
testpic	∞	∞	∞	∞
balcony	58.92	55.65	62.63	59.07

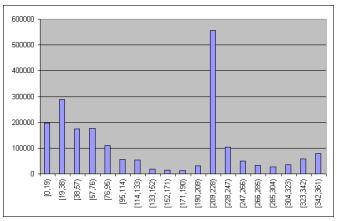
Table 6: PSNR of the recovered RGB channels after forward & inverse transform.

Figure 18 shows the resulting histograms of the test images. In the case of a black and white image as Testpic, the Hue and Saturation channels are always 0, and the Value channel corresponds with any R,G,B or Y channel.

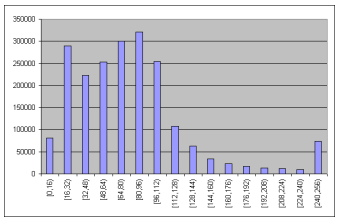
Though the results of table 6, is has been observed that this color domain has an important handicap. It is very sensitive to quantization. If no quantization is applied, just with the rounding errors quite good PSNR is obtained. But in the case that quantization is applied any change in the recovered values can produce large differences in the originals RGB pixels. In table 7 the resultant PSNR of the set of images is depicted when they are converted from RGB to HSV, taken to frequency domain by DCT transformation and its coefficients are quantized with a QP of 18.

PSNR (dBs)				
Image	R'	G'	B'	$R'G'B'$
freeway	51.01	48.72	51.14	50.30
plants	47.00	50.27	48.09	48.45
boursorma	50.61	46.13	51.49	49.41
testpic	56.47	56.47	56.47	56.47
balcony	51.40	50.55	54.09	52.01

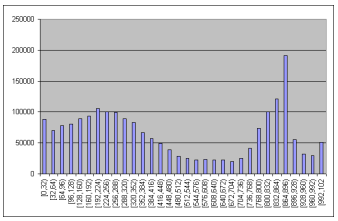
Table 7: PSNR of the recovered RGB channels affected by quantization.



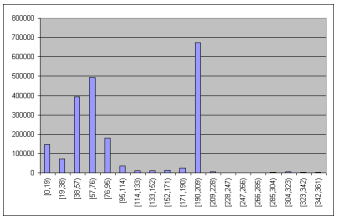
(a) Freeway Channel H



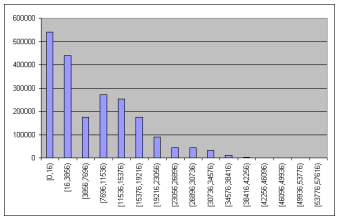
(b) Freeway Channel S



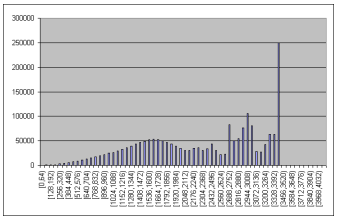
(c) Freeway Channel V



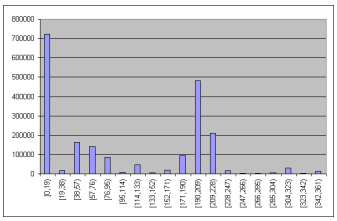
(d) Plants Channel H



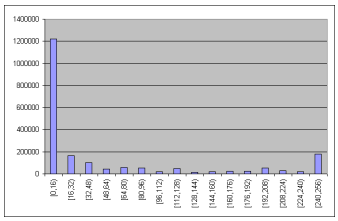
(e) Plants Channel S



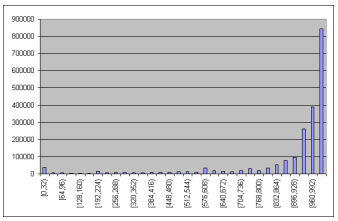
(f) Plants Channel V



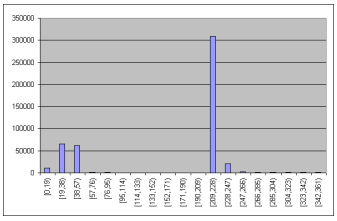
(g) Boursorma Channel H



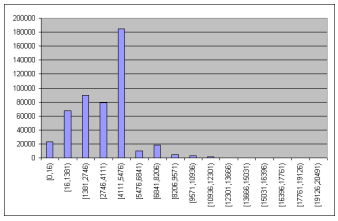
(h) Boursorma Channel S



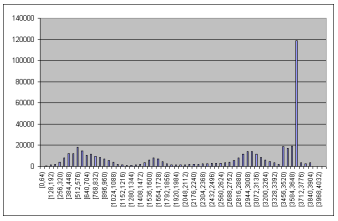
(i) Boursorma Channel V



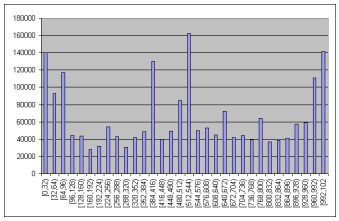
(j) Balcony Channel H



(k) Balcony Channel S



(l) Balcony Channel V



(m) Testpic Channel V

Figure 18: Histograms of the test set in HSV.

5 Codec Scheme

In this section it will be presented the EVC scheme proposed in this thesis. Figure 19 presents the block diagram of the encoder side. It components blocks will be explained in the next chapters.

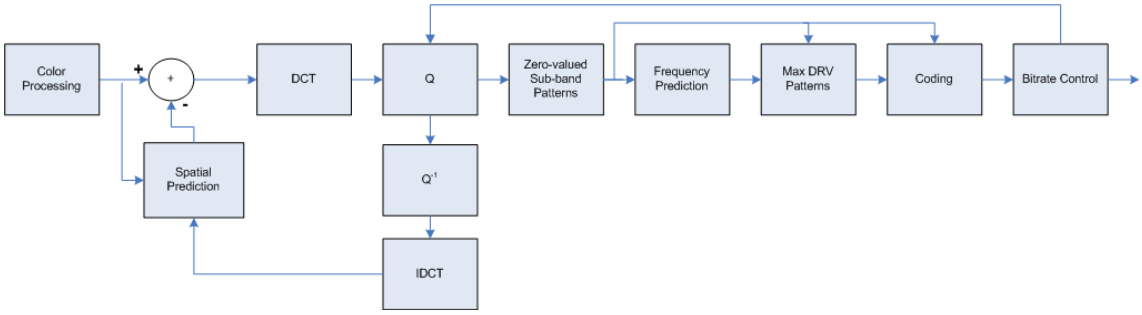


Figure 19: Encoder Block Diagram

The color processing block is used to convert the original RGB signals into another color domain. In this thesis some different color spaces have been investigated in section 4 and used for different tests, although, YUV color space has been finally selected as the scenario to use since the best results in image quality and compression ratio have been obtained using it.

Rearding the tools used to develope the CODEC model, the programing language C has been used. The back-end of the image compression framework is implemented in a modular fashion. For each step of the compression chain different interchangeable modules are provided, whose output is directed to the next module. This allows easy experimentation with different approaches for each of the steps.

The implemented framework mainly uses the Philips File Standard for Pictorial Data (PFSPD) to read and write image files to the hard disk, but is not limited to it. The PFSPD file format allows easy conversion between different color spaces and is the main format used in related projects. For a comfortable access to PFSPD files, a C library is provided by [15]. To read/write an image from/to a file, the C program accesses the CPFSPD API, which will then handle the actual file access. The PFSPD format does not only allow the storage of single image, image sequences are supported as well. This allows easy processing of complete video sequences.

In this thesis the examined compression scheme is block based. Therefore, it is advantageous to create complex data types for enabling an easy exchange of data blocks and codec syntax information between the modules. Since video data is composed by different signals, one per each plane (Y,U,V) is convenient to define a data structure for a video frame containing each plane separely. The abstraction levels implemented in the used framework are depicted in Figure 20.

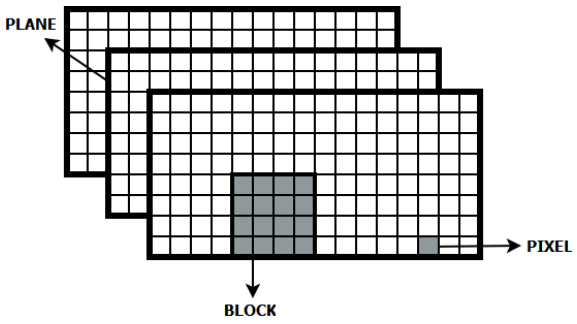


Figure 20: Video frame abstraction

The size of a block is 4×4 pixels square shaped. The Block data type is furthermore implemented as an integer array,since there are several types of blocks for different data types (DCT coefficients, image pixels, prediction residue, etc.) and they may be signed but always integer. These blocks can be share for the different modules. On Figure 21 the correspondent decoder side is depicted.

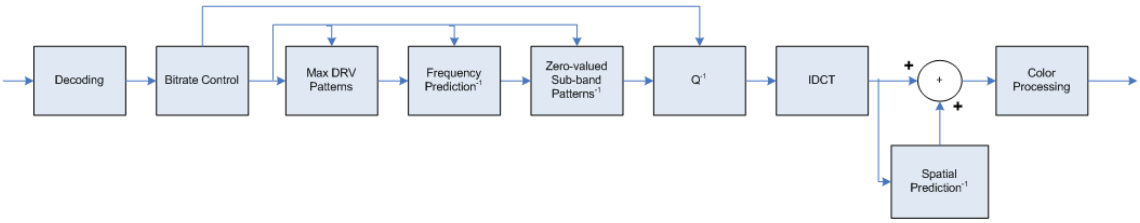


Figure 21: Decoder Block Diagram

Notice that all the explicative graphics presented along the following sections that show results of different tests are obtained for an specific scenario where the Quantization Parameter (QP) introduced in next section is always 18.

5.1 Prediction in Spatial Domain

Intra-only video coding is a widely used coding method in professional and surveillance video applications. The restriction on using only intra-frame compression is partly due to its ease of editing and partly due to the significant amount of computational complexity required by motion estimation, resulting in complex real-time video systems. In the H.264/AVC standardization process [17] the compression performance of intra coding was significantly improved by the adoption of spatial prediction in intra frames, which have permitted the H.264/AVC coder to obtain a higher compression gain with respect to the previous coding standards, like JPEG2000 (Cho et al., 2007). The pixels of the current block are predicted using the reconstructed pixels of neighboring blocks, interpolated along different orientations (Cappellari and Mian, 2004). In the first version of the H.264/AVC standard (Joint Video Team, 2002), the spatial prediction is limited to either blocks of 4×4 pixels or macroblock sized regions of 16×16 pixels. In the FRExt of the standard [18], blocks of 8×8 pixels are also considered. As a consequence, the computational complexity of an exhaustive rate-distortion optimization significantly increases because of the number of different partitioning modes and prediction directions. In order to overcome this problem, a wide variety of complexity reduction strategies, together with the introduction of novel hardware accelerators, have been proposed in literature. Due to all this complexity and the limitations of an embedded system, in this thesis a simpler version of the H.264 spatial prediction is implemented as a prior step to the DCT transform.

The input video frames are partitioned into blocks of 4×4 pixels. As it was previously introduced, the pixels used as predictors are those at the boundary of the neighboring blocks. The amount of them used each time depends on the position of the block within the image. There are five possibilities as depicted in Figure 22, which will also determine which prediction modes are available for a certain block.

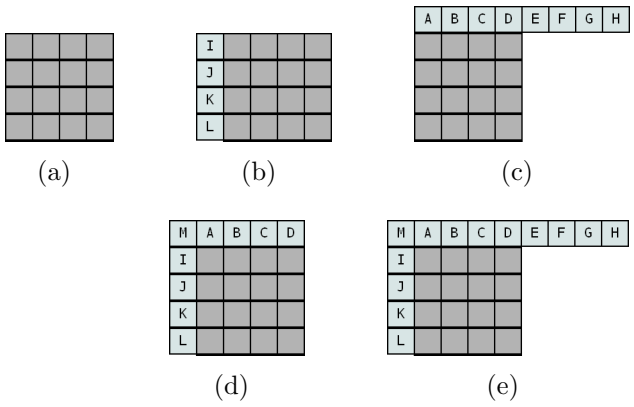


Figure 22: Pixels used as predictors: 22(a)First block in the image,22(b)blocks in the first line,22(c)blocks in the first column but not first line,22(d) blocks in the last column but not in the first line,22(e)blocks in any other position.

The pixels used as a predictor are not the original ones, but the reconstructed ones. This is done because if in the encoder original pixels are used as predictors, then in the decoder, these pixels will no be available. In the decoder side what is present are the reconstructed pixels of the previous blocks after all the decoding chain. Hence, there is a local decoder inside the encoder, see Figure 19 of section 5.

There are 9 possible prediction modes. More precisely, each candidate predictor is computed from the neighboring pixels of the upper and the left blocks, for the general case of 22(e), interpolated along an assigned spatial direction, which depends on the prediction mode. These possible spatial prediction modes are depicted in Figure 23. Equations from 37 to 46 are used for calculating the final predictor for each spatial prediction mode.

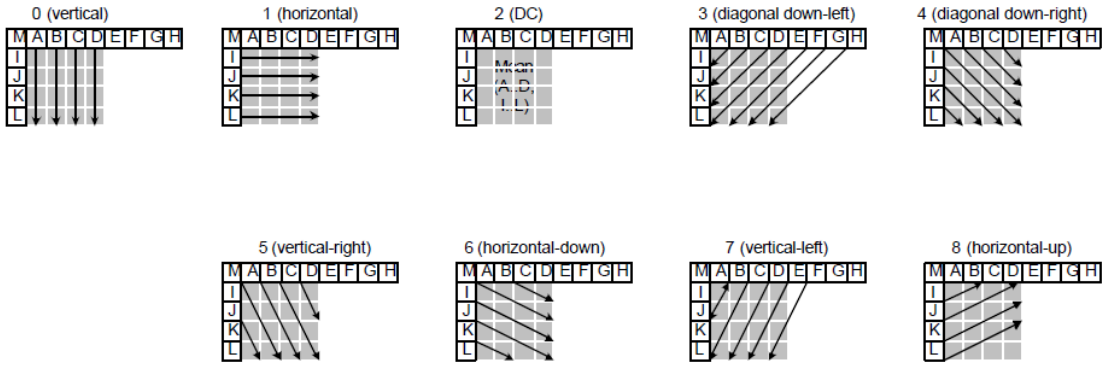
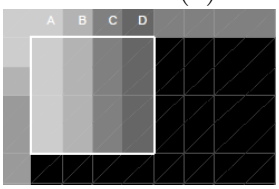


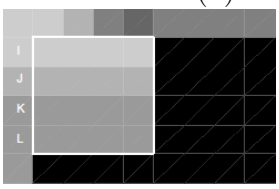
Figure 23: Spatial Prediction Modes.

Vertical (0)



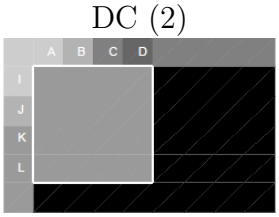
$$
\begin{aligned}
P_{i,0} &= A, \ \forall\ 0 <= i < 4. \\
P_{i,1} &= B, \ \forall\ 0 <= i < 4. \\
P_{i,2} &= C, \ \forall\ 0 <= i < 4. \\
P_{i,3} &= D, \ \forall\ 0 <= i < 4.
\end{aligned} \tag{37}
$$

Horizontal (1)



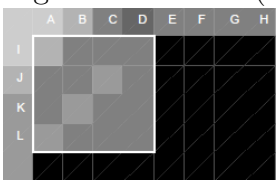
$$
\begin{aligned}
P_{0,j} &= I, \ \forall\ 0 <= j < 4. \\
P_{1,j} &= J, \ \forall\ 0 <= j < 4. \\
P_{2,j} &= K, \ \forall\ 0 <= j < 4. \\
P_{3,j} &= D, \ \forall\ 0 <= j < 4.
\end{aligned} \tag{38}
$$

DC (2)



$$
\begin{aligned}
P_{i,j} = (A + B + C + D + I + J + K + L) >> 3, \\
\forall\ 0 <= i, j < 4.
\end{aligned} \tag{39}
$$

Diagonal down-left (3)



$$
\begin{aligned}
P_{0,3} = P_{1,2} = P_{2,1} = P_{3,0} &= (D + 2 \times E + F + 2) >> 2. \\
P_{0,2} = P_{1,1} = P_{2,0} &= (C + 2 \times D + E + 2) >> 2. \\
P_{1,0} = P_{0,1} &= (B + 2 \times C + D + 2) >> 2. \\
P_{0,0} &= (A + 2 \times B + C + 2) >> 2. \\
P_{3,1} = P_{2,2} = P_{1,3} &= (E + 2 \times F + G + 2) >> 2. \\
P_{3,2} = P_{2,3} &= (F + 2 \times G + H + 2) >> 2. \\
P_{3,3} &= (G + 3 \times H + 2) >> 2.
\end{aligned} \tag{40}
$$

Diagonal down-right (4)

$$
\begin{aligned}
P_{0,0} = P_{1,1} = P_{2,2} = P_{3,3} &= (I + 2 \times M + A + 2) >> 2. \\
P_{0,1} = P_{1,2} = P_{2,3} &= (M + 2 \times A + B + 2) >> 2. \\
P_{0,2} = P_{1,3} &= (A + 2 \times B + C + 2) >> 2. \\
P_{0,3} &= (B + 2 \times C + D + 2) >> 2. \\
P_{1,0} = P_{2,1} = P_{3,2} &= (M + 2 \times I + J + 2) >> 2. \\
P_{2,0} = P_{3,1} &= (I + 2 \times J + K + 2) >> 2. \\
P_{3,0} &= (J + 2 \times K + L + 2) >> 2.
\end{aligned} \tag{41}
$$

Vertical right (5)

$$
\begin{aligned}
P_{0,0} = P_{2,1} &= (M + A + 1) >> 1. \\
P_{0,1} = P_{2,2} &= (A + B + 1) >> 1. \\
P_{0,2} = P_{2,3} &= (B + C + 1) >> 1. \\
P_{0,3} &= (C + D + 1) >> 1. \\
P_{1,0} = P_{3,1} &= (I + 2 \times M + A + 2) >> 2. \\
P_{1,1} = P_{3,2} &= (M + 2 \times A + B + 2) >> 2. \\
P_{1,2} = P_{3,3} &= (A + 2 \times B + C + 2) >> 2. \\
P_{1,3} &= (B + 2 \times C + D + 2) >> 2. \\
P_{2,0} &= (M + 2 \times I + J + 2) >> 2. \\
P_{3,0} &= (J + 2 \times K + L + 2) >> 2.
\end{aligned} \tag{42}
$$

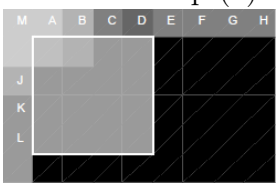
Horizontal down (6)

$$
\begin{aligned}
P_{1,0} = P_{2,2} &= (J + I + 1) >> 1. \\
P_{2,0} = P_{3,2} &= (J + K + 1) >> 1. \\
P_{3,0} &= (L + K + 1) >> 1. \\
P_{0,1} = P_{1,3} &= (A + 2 \times M + I + 2) >> 2. \\
P_{1,1} = P_{2,3} &= (M + 2 \times I + J + 2) >> 2. \\
P_{2,1} = P_{3,3} &= (I + 2 \times J + K + 2) >> 2. \\
P_{3,1} &= (J + 2 \times K + L + 2) >> 2. \\
P_{0,2} &= (M + 2 \times A + B + 2) >> 2. \\
P_{0,3} &= (A + 2 \times B + C + 2) >> 2.
\end{aligned} \tag{43}
$$

Vertical left (7)

$$
\begin{aligned}
P_{0,3} = P_{2,2} &= (D + E + 1) >> 1. \\
P_{0,2} = P_{2,1} &= (C + D + 1) >> 1. \\
P_{0,1} = P_{2,0} &= (B + C + 1) >> 1. \\
P_{1,1} = P_{3,0} &= (B + 2 \times C + D + 2) >> 2. \\
P_{1,2} = P_{3,1} &= (C + 2 \times D + E + 2) >> 2. \\
P_{1,3} = P_{3,2} &= (D + 2 \times E + F + 2) >> 2. \\
P_{0,0} &= (A + B + 1) >> 1. \\
P_{1,0} &= (A + 2 \times B + C + 2) >> 2. \\
P_{2,3} &= (E + F + 1) >> 1. \\
P_{3,3} &= (E + 2 \times F + G + 2) >> 2.
\end{aligned} \tag{44}
$$

Horizontal up (8)



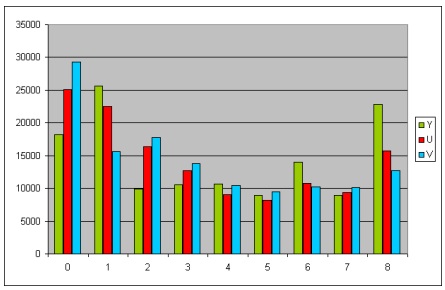
$$
\begin{aligned}
P_{3,0} = P_{2,2} = P_{3,1} = P_{3,2} = P_{3,3} = P_{2,3} &= L. \\
P_{1,2} = P_{2,0} &= (K + L + 1) >> 1. \\
P_{1,0} = P_{0,2} &= (J + K + 1) >> 1. \\
P_{2,1} = P_{1,3} &= (K + 3 \times L + 2) >> 2. \\
P_{1,1} = P_{0,3} &= (J + 2 \times K + L + 2) >> 2. \\
P_{0,0} &= (I + J + 1) >> 1. \\
P_{0,1} &= (I + 2 \times J + K + 2) >> 2.
\end{aligned} \tag{45}
$$

After the calculation of the 9 possible predictors, H.264 takes the decision of which will be finally used depending on the Sum of Absolute Errors (SAE) parameter. In Equation 46 this parameter is presented. It is calculated by the addition of the absolute values of all the differences between each pixel in the current block and the same pixel in the predictor. However, little better results were achieve in this thesis using a Modified SAE (MSAE)(Equation 47) instead of SAE as decision parameter.

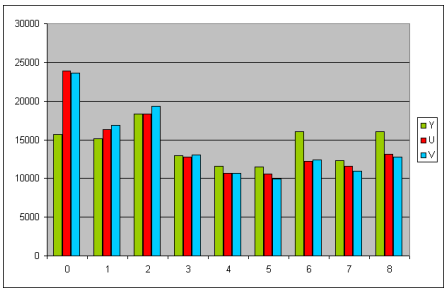
$$SAE_m = \sum_{i=0}^{15} \sum_{j=0}^{15} \left| C_{i,j} - P_{i,j}^m \right| \tag{46}$$

$$MSAE_m = \sqrt{\sum_{i=0}^{15} \sum_{j=0}^{15} (C_{i,j} - P_{i,j}^m)^2} \tag{47}$$

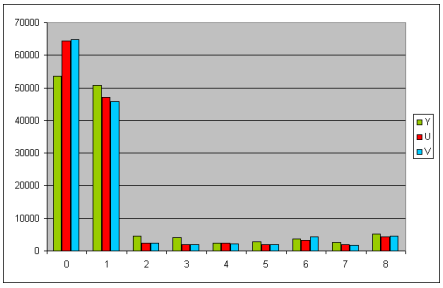
The m index indicates the prediction mode. Finally the predictor which results in the minimum MSAE will be chosen, and the residual information between $C_{i,j}$ and $P_{i,j}^m$, will be send to the next step in the coding chain together with the chosen prediction mode m. For various test pictures, Figure 24 depicts a histogram indicating the amount of times that a particular mode is selected.



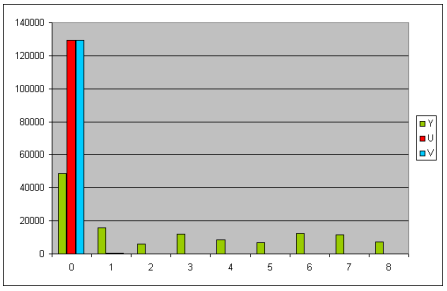
(a) freeway



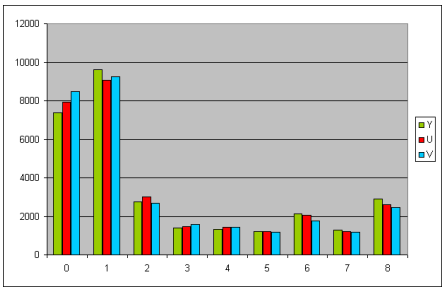
(b) plants



(c) boursorma



(d) testpic



(e) balcony

Figure 24: Distribution of spatial prediction modes

There are 9 possible directions or possible spatial prediction modes, which requires 4 bits to literally code them. H.264 uses a different technique, which on average requires less bits. The numbers of the different directions are ordered from most probable for the Vertical mode (0), to the less probable Horizontal-up (8). Taking advantage of this, H.264 calculates the most probable mode, which takes its value following Equation 48.

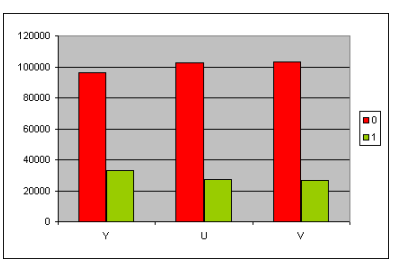
$$most\ probable\ mode = min(m_{leftblock}, m_{upper\ block}) \tag{48}$$

If the actual calculated prediction mode for the current block matches the most probable mode then a flag called *most_probable_mode* is set to 1. If this is not the case, then the flag *most_probable_mode* is set to 0 and the spatial prediction mode is sent requiring only 3 bits, because the most probable mode has been already discarded so there are only 8 of the original 9 modes remaining.

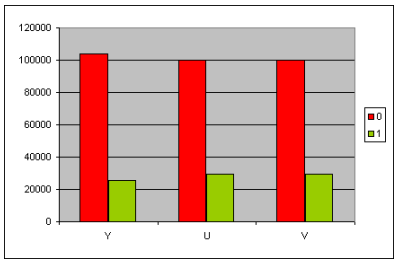
This method has been tested, but a slight modification has been introduced, which proved to work better for different kind of video content and it requires only a single

comparison function. Instead of calculating the most probable mode as H.264 does, it uses the mode of the previous block. The use of the flag *most_probable_mode* is avoided, and in stead of this it uses a flag called *previous_spmode* which works the same way as *most_probable_mode*, but comparing the actual prediction mode with the mode of the previous block. In Figure 25 the number of times *previous_spmode* takes each value is depicted. In order to be able to code the prediction mode with three bits, for the situation that the actual one and the previous one do not match, in the case that the actual prediction mode is larger than the previous, it is decreased one unity. In the decoder if *previous_spmode* is 0, then the prediction mode is received. If it is larger or equal than the previous mode it is added one unity to become the original value again.

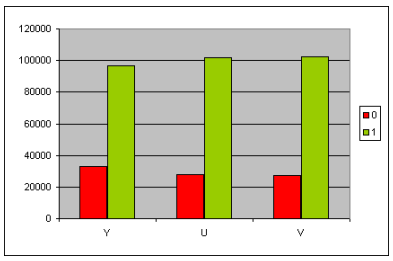
Summarizing, after calculating the prediction mode of the current block minimizing the MSAE, if it matches the prediction mode used for the previous block, it requires only a single bit to send the mode activating *previous_spmode*. If it does not match, then four bits are needed, one for *previous_spmode*, and three to code the prediction mode.



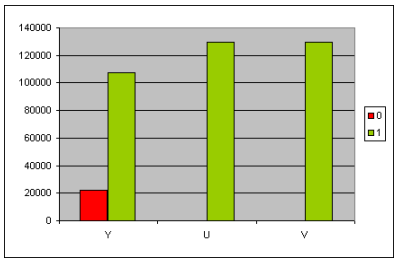
(a) freeway



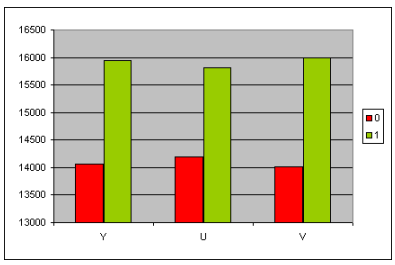
(b) plants



(c) boursorma



(d) testpic



(e) balcony

Figure 25: Distribution of *previous_spmode* flag

5.2 DCT Transformation & Quantization

In an H.264/AVC codec, a picture is divided into macroblocks, which are further divided into blocks of either 4×4 or 8×8. These blocks are transformed and quantized prior to entropy coding. At the decoder, the inverse operation takes place involving inverse entropy, inverse quantization and inverse transform.

This section elaborates on the 4×4 Discrete Cosine Transform (DCT) and associated quantization as deployed in H.264 [17]. Purpose of this section is to introduce the DCT, which is later used in a transform based video compression scheme. The transform is not an exact DCT transform, but a scaled approximation of the 4×4 DCT with the advantage of using simple integer arithmetic. The required normalization step is incorporated into forward and inverse quantization operations.

5.2.1 Forward Transform and Quantization Process

The basic 4×4 transform used in H.264 is a scaled approximate DCT. The transform and quantization processes are structured such that computational complexity is minimized. This is achieved by reorganizing the transform into a core part and a scaling part.

In a common case, an original image block of pixels would be processed by a 2D-DCT and then quantized, dividing by a quantization step Q_{step} and rounding the result, as depicted in 26(a). In H.264, the DCT transform is divided in two different processes. On one hand a core transform C_f and on the other hand a scaling matrix (S_f). Then, quantization process is up-scaled by a constant 2^{15}, compensated and finally rounded as shown in 26(b). Finally, the scaling and quantization step merge in M_f (26(c)).

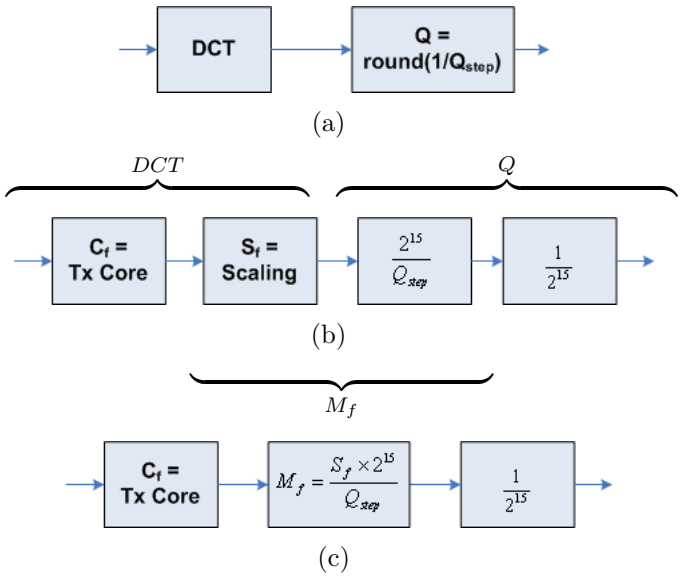


Figure 26: Development of forward transform and quantization

5.2.2 Rescaling and Inverse Transform Process

To recover the original data, inverse quantization and 2D Inverse Discrete Cosine Transformation (IDCT) are required (27(a)). Similar as in the forward transform, the IDCT is divided in a core transformation C_i and a scaling matrix S_i. The inverse quantization matrix is scaled by the constant 2^6 and rescaled and rounded at the end of the chain (27(b)). Last step is to combine the re-scaling process and S_i into V_i (27(b)).

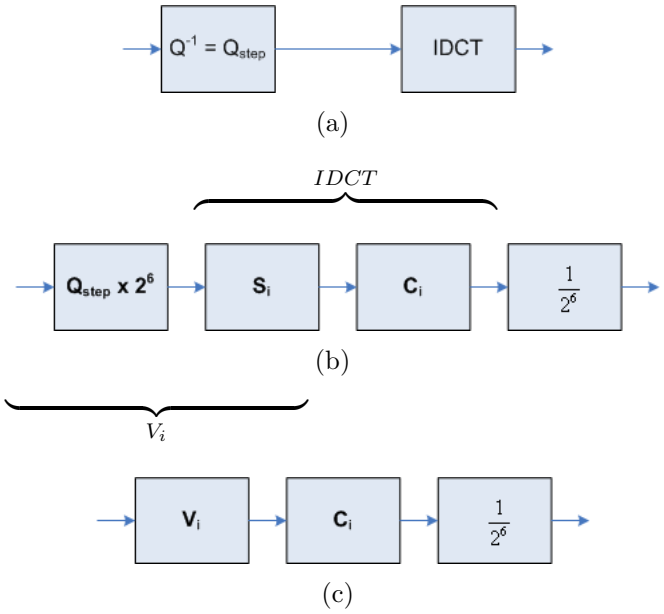


Figure 27: Development of the rescaling and inverse transform proces

5.2.3 Developing of C_f and S_f 4×4 matrixes

Let A be the 4×4 DCT transform matrix, then the transformed coefficients Y of the original pixels X are:

$$Y = A \times X \times A^T \tag{49}$$

The rows of A are orthogonal and have unit norms (i.e. the rows are orthonormal):

$$\begin{bmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{bmatrix}, \quad \begin{aligned} a &= 1/2 \\ b &= \sqrt{1/2}\cos(\Pi/8) = 0.6532... \\ c &= \sqrt{1/2}\cos(3\Pi/8) = 0.2406... \end{aligned}$$

Calculation of Equation 49 on a practical processor requires approximation of the irrational numbers b and c. A fixed-point approximation is equivalent to scaling

each row of A and rounding to the nearest integer. Following this method:

$$C_f = round(2.5 \times A) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

This approximation is chosen to minimize the complexity of implementing the transform (multiplication by C_f requires only additions and binary shifts) while maintaining good compression performance. The rows of C_f have different norms and C_f needs to be orthonormal. To do so,all the values c_{ij} in row r are divided by $\frac{1}{\sqrt{\sum_j c_{rj}^2}}$ resulting in R_f:

$$R_f = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \\ 1/2 & 1/2 & 1/2 & 1/2 \\ 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} & 1/\sqrt{10} \end{bmatrix}$$

A new A' is defined as $A' = C_f \cdot R_f$. Notice that $\cdot$ denotes element-by-element multiplication. Using A' Equation 49 becomes:

$$\begin{aligned} Y &= A' \times X \times A'^T = [C_f \cdot R_f] \times X \times [C_f^T \cdot R_f^T] \\ &= [C_f \times X \times C_f^T] \cdot [R_f \cdot R_f^T] \\ &= [C_f \times X \times C_f^T] \cdot S_f \end{aligned} \tag{50}$$

where

$$S_f = R_f \cdot R_f^T = \begin{bmatrix} 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \\ 1/4 & 1/2\sqrt{10} & 1/4 & 1/2\sqrt{10} \\ 1/2\sqrt{10} & 1/10 & 1/2\sqrt{10} & 1/10 \end{bmatrix}$$

5.2.4 Developing of C_i and S_i 4×4 matrixes

Consider A, the same 4×4 DCT transform matrix in the previous section. A is also the IDCT transform matrix as $A^{-1} = A$. Then the the recovered pixels Z are:

$$Z = A^T \times Y \times A \tag{51}$$

Scaling each row of A and rounding to the nearest 0.5:

$$C_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

The rows of Ci are orthogonal but have non-unit norms. To restore orthonormality all values c_{ij} in row r are multiply by $\frac{1}{\sqrt{\sum_j c_{rj}^2}}$ resulting in R_i:

$$R_i = \begin{bmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ \sqrt{2/5} & \sqrt{2/5} & \sqrt{2/5} & \sqrt{2/5} \\ 1/2 & 1/2 & 1/2 & 1/2 \\ \sqrt{2/5} & \sqrt{2/5} & \sqrt{2/5} & \sqrt{2/5} \end{bmatrix}$$

The new A'' and 2D inverse transform becomes:

$$A'' = C_i \cdot R_i \tag{52}$$

$$\begin{aligned} Z &= A''^T \times Y \times A'' = [C_i^T \cdot R_i^T] \times Y \times [C_i \cdot R_i] \\ &= [C_i^T] \times [Y \cdot R_i^T \cdot R_i] \times [C_i] \\ &= [C_i^T] \times [Y \cdot S_i] \times [C_i] \end{aligned} \tag{53}$$

where

$$S_i = R_i^T \cdot R_i = \begin{bmatrix} 1/4 & 1/\sqrt{10} & 1/4 & 1/\sqrt{10} \\ 1/\sqrt{10} & 2/5 & 1/\sqrt{10} & 2/5 \\ 1/4 & 1/\sqrt{10} & 1/4 & 1/\sqrt{10} \\ 1/\sqrt{10} & 2/5 & 1/\sqrt{10} & 2/5 \end{bmatrix}$$

5.2.5 Developing of rescaling matrix V_i

In the block diagram of 27(c) a block V_i is introduced, which is defined by:

$$V_i = S_i \cdot 2^6 \cdot Q_{step} \tag{54}$$

A total of 52 Q_{step} are supported by H.264, indexed by the Quantization Parameter, QP (Table 8). The ratio between successive Q_{step} values is chosen to be $\sqrt[6]{2} = 1.2246..$ so that Q_{step} doubles in size when QP increases by 6. Any value of Q_{step} can be derived from the first 6 values in the Table 8 ($QP_0 - QP_5$) as follows:

$$Q_{step}(QP) = Q_{step}(QP\%6) \times 2^{floor(QP/6)}. \tag{55}$$

QP	0	1	2	3	4	5	6	.	12	.	16	.	48	.	51
Q_{step}	0.625	0.702	0.787	0.884	0.992	1.114	1.250	.	2.5	.	5.0	.	160	.	224

Table 8: QP and Q_{step} relation

Since the values of V_i depend on Q_{step} they also depend on QP. Q_{step} can be calculated using the first 6 values of QP. The calculation for V_i follows a similar

approach only requiring two additional arguments S_i and 2^6. From observation it becomes clear that every time QP increases by 6 Q_{step} and V_i double. In Table 9 the first 6 matrixes V_i are shown:

QP	$Q_{step} \times 2^6$	$V_i = round(S_i \times Q_{step} \times 2^6)$
0	40	$\begin{bmatrix} 10 & 13 & 10 & 13 \\ 13 & 16 & 13 & 16 \\ 10 & 13 & 10 & 13 \\ 13 & 16 & 13 & 16 \end{bmatrix}$
1	44.898	$\begin{bmatrix} 11 & 14 & 11 & 14 \\ 14 & 18 & 14 & 18 \\ 11 & 14 & 11 & 14 \\ 14 & 18 & 14 & 18 \end{bmatrix}$
2	50.397	$\begin{bmatrix} 13 & 16 & 13 & 16 \\ 16 & 20 & 16 & 20 \\ 13 & 16 & 13 & 16 \\ 16 & 20 & 16 & 20 \end{bmatrix}$
3	56.569	$\begin{bmatrix} 14 & 18 & 14 & 18 \\ 18 & 23 & 18 & 23 \\ 14 & 18 & 14 & 18 \\ 18 & 23 & 18 & 23 \end{bmatrix}$
4	63.496	$\begin{bmatrix} 16 & 20 & 16 & 20 \\ 20 & 25 & 20 & 25 \\ 16 & 20 & 16 & 20 \\ 20 & 25 & 20 & 25 \end{bmatrix}$
5	71.272	$\begin{bmatrix} 18 & 23 & 18 & 23 \\ 23 & 29 & 23 & 29 \\ 18 & 23 & 18 & 23 \\ 23 & 29 & 23 & 29 \end{bmatrix}$

Table 9: First matrixes for V_i

Note that there are only three unique values in each matrix V_i . These three values are defined as a table of values v in the H.264 standard, for $QP = 0$ to $QP = 5$:

QP	$v(r,0)$ V_i positions $(0,0),(0,2),(2,0),(2,2)$	$v(r,1)$ V_i positions $(1,1),(1,3),(3,1),(3,3)$	$v(r,2)$ V_i remaining positions
0	10	16	13
1	11	18	14
2	13	20	16
3	14	23	18
4	16	25	20
5	18	29	23

Table 10: $v(r,n)$ values

Hence for QP values from 0 to 5, V_i is obtained as:

$$V_i = \begin{bmatrix} v(QP,0) & v(QP,2) & v(QP,0) & v(QP,2) \\ v(QP,2) & v(QP,1) & v(QP,2) & v(QP,1) \\ v(QP,0) & v(QP,2) & v(QP,0) & v(QP,2) \\ v(QP,2) & v(QP,1) & v(QP,2) & v(QP,1) \end{bmatrix}, V_i = v(QP,n) \tag{56}$$

Where $v(r,n)$ is row r, column n of v. For larger values of QP ($QP > 5$):

$$V_i = v(QP\%6, n) \times 2^{floor(QP/6)} \tag{57}$$

5.2.6 Developing M_f matrix

Combining first M_f definition in 26(c) and Equation 54, Equation 58 is obtained:

$$M_f = round(\frac{S_f \cdot S_i \cdot 2^{21}}{V_i}) \tag{58}$$

Notice the $round()$ function because S_i and S_f are irrational numbers. As S_i and S_f are known, first it can be calculated:

$$S_f \cdot S_i \cdot 2^{21} = \begin{bmatrix} 131072 & 104857.6 & 131072 & 104857.6 \\ 104857.6 & 83886.1 & 104857.6 & 83886.1 \\ 131072 & 104857.6 & 131072 & 104857.6 \\ 104857.6 & 83886.1 & 104857.6 & 83886.1 \end{bmatrix} \tag{59}$$

Then M_f can be calculated following Table 11, similar to V_i:

QP	$m(r,0)$ M_f positions $(0,0),(0,2),(2,0),(2,2)$	$m(r,0)$ $M_f$positions $(1,1),(1,3),(3,1),(3,3)$	$m(r,2)$ M_f remaining positions
0	13107	5243	8066
1	11916	4660	7490
2	10082	4194	6554
3	9362	3647	5825
4	8192	3355	5243
5	7282	2893	4559

Table 11: $m(r,n)$ values

Similar to V_i a matrix for M_f is defined by:

$$M_f = \begin{bmatrix} m(QP,0) & m(QP,2) & m(QP,0) & m(QP,2) \\ m(QP,2) & m(QP,1) & m(QP,2) & m(QP,1) \\ m(QP,0) & m(QP,2) & m(QP,0) & m(QP,2) \\ m(QP,2) & m(QP,1) & m(QP,2) & m(QP,1) \end{bmatrix}, M_f = m(QP,n) \tag{60}$$

$$M_f = m(QP\%6, n)/2^{floor(QP/6)} \tag{61}$$

Where $m(r,n)$ is row r, column n of m.

5.2.7 Complete forward and Inverse Transformations

Summarizing and combining the results of previous sections results in a Forward transform, scaling and quantization process:

$$Y = round([C_f \times Y \times C_f^T] \cdot m(QP\%6, n)/2^{floor(QP/6)} \times \frac{1}{2^{15}}) \tag{62}$$

Inverse transform and scaling process:

$$Z = round([C_i^T] \times [Y \cdot v(QP\%6, n) \times 2^{floor(QP/6)}] \times [C_i] \times \frac{1}{2^6}) \tag{63}$$

5.3 Zero-valued sub-bands patterns

The objective of this technique is coding the zero-valued DCT coefficients in an efficient manner. In an early phase,this concept was thought for graphic-based video data [16], because pixel values in spatial domain tend to be repetitive of nature, resulting in the presence of zero-valued sub-bands in the 2D-DCT block. As a second step this method can be extended to natural video, where these structures are not that common. But after the quantization step, some coefficients tend to be zero, mostly the ones of high frequencies, resulting in zero-valued sub-bands patterns that can be exploited. In order to do so, a set of 19 patterns containing those zero-valued sub-bands have been developed. Studying different video material those patterns which are repeated the most are taken as DCT blocks classification. These patterns take into account possible symmetries that can occurred in a certain block allowing to reduce the number of coefficients that need to be transmit. On the other hand, not just symmetries are used, but also certain coefficients distributions which happen significantly often are taken into account. The position of the Zero-valued sub-bans patterns block within the codec chain is shown in figure:

Figure 28: Zero sub-band pattern selection after DCT and Quantization

The set of exact symmetries results in 8 modes. The names in capital letters will be used from now on to reference each pattern:

- Uniformity, UNISAD
- Vertical and Horizontal Symmetry, VERTUNISAD and HORIZUNISAD
- Vertical and Horizontal Uniformity, VERTSYMSAD and HORIZSYMSAD
- Diagonal and Anti-diagonal Symmetry, SYMSAD and ANTISYMSAD
- Mirror Symmetry, MIRRORSYMSAD

The set of patterns is extended by adding additional modes, which are not based on symmetry properties. From observation of video data there have been selected ten more patterns where zero-valued coefficients occurs in a row, column or combination of them within the 4×4 DCT block:

- All zeros in Row 3, NOSYMROW3
- All zeros in Column 3, NOSYMCOL3
- All zeros in Row 2, NOSYMROW2

- All zeros in Column2, NOSYMCOL2
- All zeros in Column 2 and 3, NOSYMCOL23
- All zeros in Row 2 and Row 3, NOSYMROW23
- All zeros in Row 3 and Column 2, NOSYMROW3COL2
- All zeros in Row 3 and Column 3, NOSYMROW3COL3
- All zeros in Row 2 and 3 and Column 2 and 3, NOSYMROWCOL23
- All zeros in Lower right 2x2 corner of the block, NOSYMLWRCOR

No SYMmetry (NOSYM) is the last pattern that is used to indicate that no pattern was found in the block, so all the coefficients will be taken into account.

5.3.1 Uniformity mode

This mode is selected when all the values in the complete 4×4 block are equal, implying that the corresponding DCT block will have a unique non-zero coefficient at the DC position, while all others sub-bands are zero. This situation occurs quite frequently in graphic data, or in flat regions in natural video, which may due to quantization.

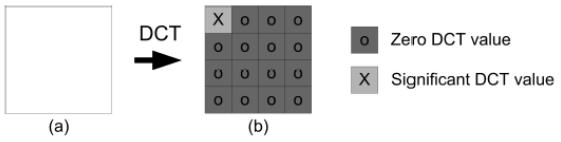


Figure 29: (a) Pattern in Spatial Domain, (b) Pattern in Frequency Domain

5.3.2 Vertical Symmetry mode

When the block in spatial domain shows symmetry along an horizontal axis in the middle of the 4×4 block as it is seen in Figure 30, this results in non-zero valued sub-bands located at the first and third rows of the DCT matrix.

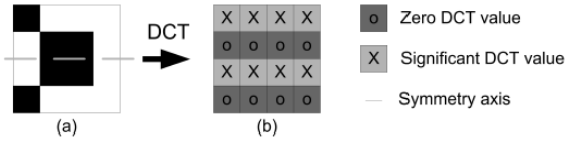


Figure 30: (a) Pattern in Spatial Domain, (b) Pattern in Frequency Domain

5.3.3 Horizontal Symmetry mode

This is similar as in Vertical Symmetry, but the symmetry axis is in vertical direction. This way the non-zero valued sub-bands are located in the first and third columns of the DCT matrix.

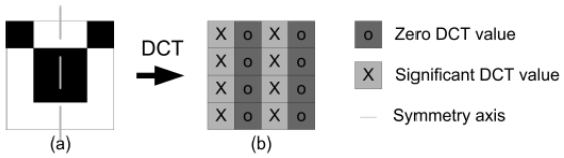


Figure 31: (a) Pattern in Spatial Domain, (b) Pattern in Frequency Domain

5.3.4 Vertical Uniform Symmetry and Horizontal Uniform Symmetry modes

These are special cases of Vertical and Horizontal Symmetry respectively, resulting in more zero-valued sub-bands. In spatial domain these blocks present a double symmetry as shown in Figures 32 and 33.

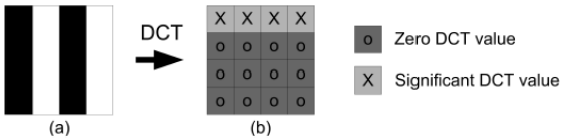


Figure 32: (a) Pattern in Spatial Domain, (b) Pattern in Frequency Domain

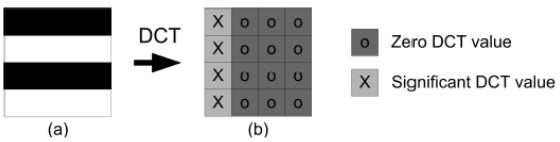


Figure 33: (a) Pattern in Spatial Domain, (b) Pattern in Frequency Domain

5.3.5 Diagonal Symmetry mode

In this mode the symmetry is obtained along the diagonal axis, i.e. the $x_{i,j}$ elements with $i = j$ as well in the spatial and transform domain. As a result, it requires only transmission of those coefficients at the diagonal and coefficients above or below the diagonal.

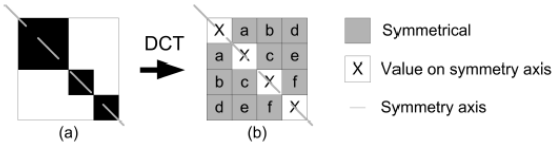


Figure 34: (a) Pattern in Spatial Domain, (b) Pattern in Frequency Domain

5.3.6 Anti-diagonal Symmetry mode

This is similar as the Diagonal Symmetry, but the symmetry axis in the spatial domain is rotated 90° clockwise, whereas the symmetry axis in the transform domain is still along the diagonal axis. As a result, there is still symmetry from the sub-band amplitude point of view, but the sign of the non-zero amplitudes is opposite, as can be seen in Figure 35.

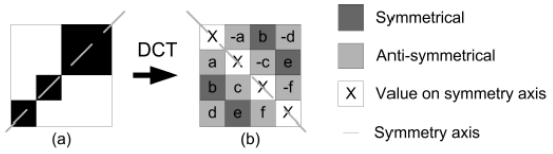


Figure 35: (a) Pattern in Spatial Domain, (b) Pattern in Frequency Domain

5.3.7 Mirror Symmetry mode

In this coding mode, the block presents a symmetry and an anti-symmetry. On one hand it presents a vertical or horizontal symmetry in one of the halves of the block, and on the other hand, the other half of the block is the mirror view of the first one. In Figure 36 it can be seen this property which derivate in a chess-pattern DCT block.

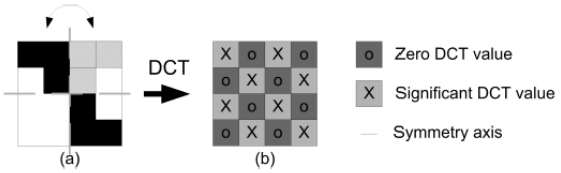


Figure 36: (a) Pattern in Spatial Domain, (b) Pattern in Frequency Domain

5.3.8 No symmetry modes

Regarding those blocks that not show an exact symmetry, it is still possible to find certain sub-bands that are zero valued due to the distribution of the pixels or due to the amount of applied quantization. In the transformed DCT block some coefficients are zero-valued resulting in an entire rows, columns or combination of them filled

with zeros. Clustering them in a pattern increases the coding efficiency. In Figure 37 it can be seen a representation of all these modes.

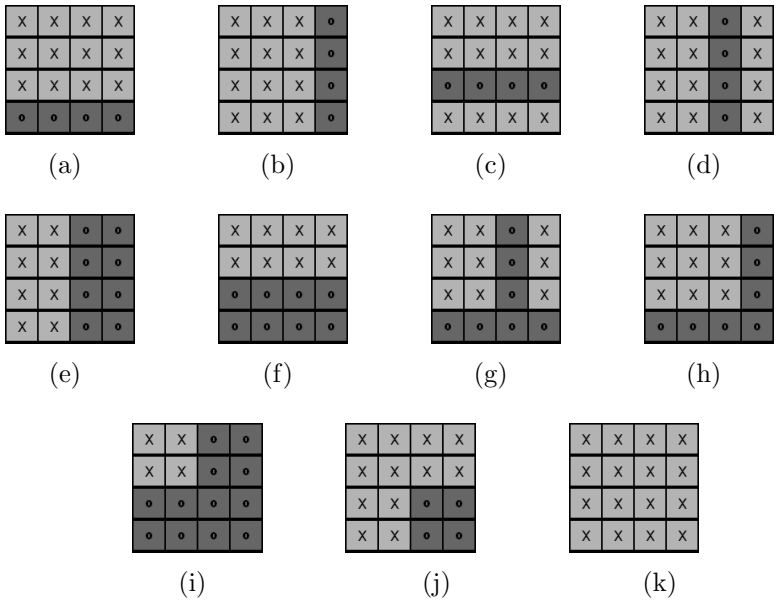


Figure 37: NOSYM Zero sub-bands patterns in DCT blocks: (a) NOSYMROW3, (b) NOSYMCOL3, (c) NOSYMROW2, (d) NOSYMCOL2, (e) NOSYMCOL23, (f) NOSYMROW23, (g) NOSYMROW3COL2, (h) NOSYMROW3COL3, (i) NOSYMROWCOL23, (j) NOSYMLWRCOR, (k) NOSYM

Selection of the coding pattern only depends on the amount of clustered zero-valued sub-bands. The coding pattern that clusters the higest amount of zero-valued sub-bands will be selected as the final coding pattern. Once a coding pattern is determined the zero-valued sub-bands covered by the coding mode are ignored in the further processing. So the selection process consist in checking every pattern in the block beginning with the UNISAD pattern since it has only one non-zero coefficient and finishing with NOSYMLWRCOR, which has only 4 zero-valued sub-bands. The process stops at the moment that a pattern fits with the DCT block. In table 12 it is shown the selection order and the amount of saved sub-bands in each mode.

In order to notify the decoder which coding pattern has been chosen, a mechanism is deployed consisting of two steps. First, the *most probable pattern*(mpp) is calculated by:

$$mpp = max(P_{prev}, P_{up}) \tag{64}$$

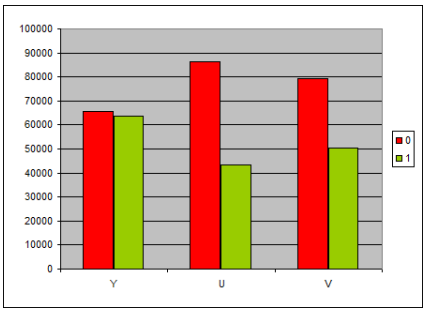
where P_{prev} is the coding pattern selected for the previous block and P_{up} is the pattern selected in the upper block. If the current selected pattern is the same than mpp, then a flag called *mpp_symmetry_mode* is activated with "1", if not then it is set to "0". *mpp_symmetry_mode* is always sent. This method exploits correlation between neighboring blocks. Since neighboring pixels often present correlation, it may be probable closed blocks have the same zero sub-bands.

Pattern	Mode	Position	Sub-bands	Pattern	Mode	Position	Sub-bands
UNISAD	0	1	15	NOSYMROW3COL2	10	11	7
HORIZUNISAD	1	2	12	SYMSAD	11	12	6
VERTUNISAD	2	3	12	ANTISYMSAD	12	13	6
NOSYMROWCOL23	3	4	12	NOSYMCOL3	13	14	4
MIRRORSYMSAD	4	5	8	NOSYMCOL2	14	15	4
HORIZSYMSAD	5	6	8	NOSYMROW3	15	16	4
VERTSYMSAD	6	7	8	NOSYMROW2	16	17	4
NOSYMCOL23	7	8	8	NOSYMLWRCOR	17	18	4
NOSYMROW23	8	9	8	NOSYM	18	19	0
NOSYMROW3COL3	9	10	7				

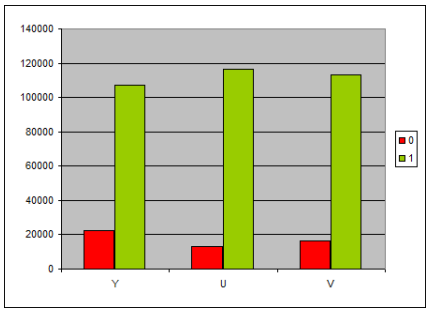
Table 12: Number value of each pattern, order of selection and sub-bands saved.

The max() function returns the worst case, as the mode index of each pattern is higher when less zero sub-bands are present. Moreover, in empiric tests this function gave better results than min() or just one of the other patterns (P_{prev},P_{up}).At the upper border of the video frame, mpp equals to P_{prev}, and for the left border of the video frame, mpp equals to P_{up}. In Figure 38 the *mpp_symmetry_mode* flag distribution is depicted.

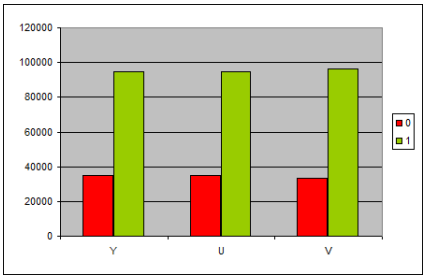
The second step is done because there are 19 possible Zero sub-band pattern modes, so it needs 5 bits to code them literally, but some codewords are wasted, since with 5 bits it is possible to code 32 codewords. For this reason, a new flag *high_symmetry_modes* is used to signal that the selected coding pattern belongs to the highest modes on Table 12, which means larger than coding pattern mode NOSYMROW3COL2 (10). This way, when *mpp_symmetry_mode* is 0, *high_symmetry_modes* is sent, being "1" when the condition is fulfilled and "0" if it is not. For the situation that *high_symmetry_modes* is "1", then the selected coding pattern mode is subtracted from SYMSAD (11) resulting in all those patterns from SYMSAD (11) to NOSYM(18) are coded with mode values from 0 to 7, which can be coded with 3 bits. For the situation that *high_symmetry_modes* is "0", the coding pattern is between UNISAD (0) and NOSYMROW3COL2(10), and it is coded with 4 bits. This way, if mpp is not equal to the actual mode, the mode is sent using *high_symmetry_modes*+3 bits, 4 bits in total, or *high_symmetry_modes*+4 bits, 5 bits total, saving 1 bit for the highest modes which also consume more bits for having more no zero sub-bands.



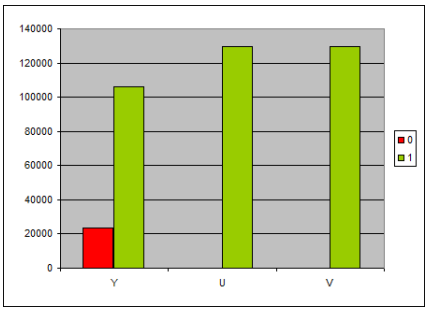
(a) freeway



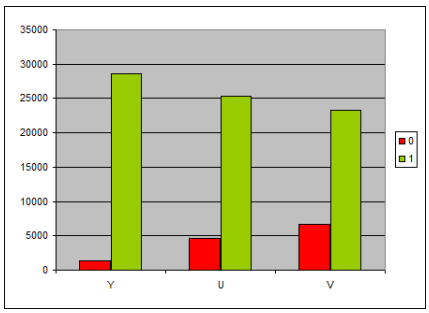
(b) plants



(c) boursorma



(d) testpic



(e) balcony

Figure 38: Distribution of *mpp_symmetry_mode* flag

5.4 Prediction in Frequency Domain

Similar to H.264 where spatial correlation between blocks of an image is exploited as discussed in subsection 5.1, it is possible to benefit from correlation between neighboring DCT blocks. Observation revealed that in neighboring DCT blocks many coefficients show similar amplitude values, which may differ in sign. Where H.264 uses various directions to calculate a predictor, prediction in the frequency domain is limited to full block subtraction or addition. The methods deployed in H.264 can not be used in the frequency domain since each coefficient represents energy at different frequency. The concept deployed in this thesis exploits the strength of coding patterns, as discussed in subsection 5.3 combined with the low-cost arithmetic operation addition and subtraction of neighboring DCT blocks.

5.4.1 DC Coefficient Prediction

Let C be a matrix of $N \times N$, which $N = 4$ being the size of the DCT block, then $c_{0,0}$ is called DC coefficient. It represents the DC component in the DCT block, and its value corresponds with the mean value of the 16 coefficients. The DC coefficient is usually the one which presents the highest amplitude in the block, since most of the video data is low-frequency based. One technique that is widely deployed is subtracting half-range to all the pixels in the spatial block before performing the DCT transform, so that the DC component of the block will be decreased since the mean value of the pixels in the block will decrease half-range. In this thesis half range subtraction is not explicitly used because spatial pixels are predicted using the H.264 methods, which implicitly result in a form of DC scaling.

On the other hand, usually DC coefficients present a high correlation. Taking into account all these facts, the DC coefficients of successive blocks are DPCM coded, where the DC coefficient of the previous block is subtracted from current DC coefficient. The DPCM operation, frequently result in a low amplitude delta DC value. The final entropy coding of the DC information can benefit from this low-dynamic range when using Dynamic Range (DR) coding in combination with Rice coding. In order to decode the DC value at the decoder side, two types of information is required. First, information concerning the dynamic range. Second, information concerning the actual differential DC value. The former information field is Rice coded because the dynamic range gradually changes over time resulting in high correlation of the dynamic range between successive DCT blocks. Because the decoder has knowledge on the DC dynamic range, the actual differential coded DC value uses the exact amount of bits. The DC DR needed to code the difference value is calculated (DR_{DC}). If it is 0, meaning the difference is 0, no DC value is sent. If it is 1, then just the sign bit is sent. For all other case, the value $2^{DR_{DC}-1}$ is subtracted. This way the first bit of the difference amplitude that is always 1 and therefore not sent, and it needs $DR_{DC} - 1$ bits to code the amplitude. Decoding requires the

inverse operation.

$$\begin{aligned} DC &= c^0_{0,0} - c^1_{0,0} \\ DR_{DC} &= DR(DC) \\ DC' &= DC - (2 << (DR_{DC} - 1)) \end{aligned} \tag{65}$$

Equation 65 depicts the calculations involved for the DC coding, where $c^0_{0,0}$ is the DC coefficient of the actual block, $c^1_{0,0}$ is the DC coefficient of the previous block, and DC is their difference. DR_{DC} is the DR needed to code DC and DC' is the value that is finally sent after removing the first 1 bit of the DC value. Notice that many times the DC coefficients of NOSYM blocks are higher since no coefficient in the block is 0 and spatial predictions may work worse on these blocks. For that reason, the calculation explained here is done separately for those blocks that are NOSYM (18) and the rest of them. When the situation that the current block is NOSYM (18), its DC coefficient is predicted from the previous NOSYM (18) block, which may not be consecutive. If a non NOSYM (18) block arrives, then its DC coefficient is predicted from the last non NOSYM (18) block, which may not be consecutive.

5.4.2 1D Frequency Prediction: Left block Predictor

The basic concept here is to perform a DPCM operation between two consecutive DCT blocks, where each sub-band of the previous block at position $x_{i,j}$ is subtracted from the sub-band at equal position in the current block. But there are some issues, when performing such a prediction since in a previous step of the coding chain, a coding pattern has been calculated for the current block. The zero-valued coefficients are not considered in the prediction process. In this way increasing the amplitude of zero-valued sub-bands covered by the coding pattern are discarded, although it is still possible to increase individual zero-valued sub-bands not covered by the coding pattern.

Moreover, DCT coefficients are signed, resulting in increased amplitude for the situation that two amplitudes with different signs are subtracted. There are two options to avoid this. The first option uses only absolute values of the non-zero sub-bands. This requires an additional sign bit for each subtracted coefficient. The second option, which is used in this project, avoids an additional sign bit using the subtraction and addition operations. The objective is to lower the dynamic range of the DCT sub-bands, which is either obtained by subtraction or addition of the predictor. Note that it is possible that non of the operations result in a lower dynamic range. In Figure 39 there is an example of the prediction concept. For simplicity reasons, the current block and also the previous block are both of type NOSYM coding pattern. As a result there are no zero-valued rows, columns or combinations of them within the DCT block, so all sub-bands are used for prediction. The results of applying the two possible predictors can be calculated in parallel following the scheme on Figure 40 and Equation 66. Notice that prior to the bit cost calculation and frequency prediction mode decision there is another block in doted line, Max DRV patterns.

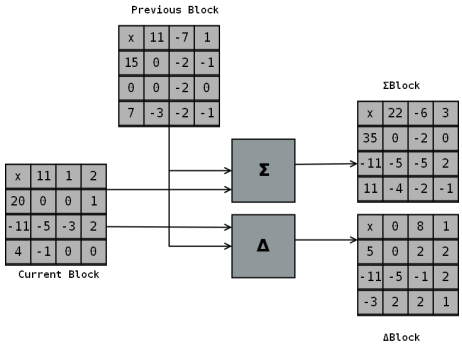


Figure 39: $\Delta Block$ and $\Sigma Block$ Predictors

This block will be elaborated in the following section, and is only active for NOSYM coding pattern blocks.

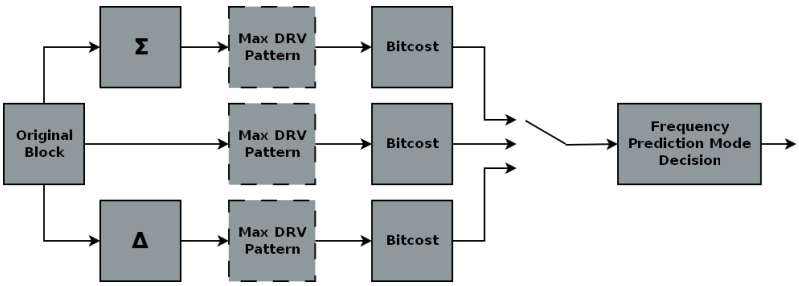


Figure 40: Frequency Prediction Chain

$$C^{\Delta}_{i,j} = C^{0}_{i,j} - C^{1}_{i,j} \quad \forall\ i,j \neq 0,0 \tag{66}$$

$$C^{\Sigma}_{i,j} = C^{0}_{i,j} + C^{1}_{i,j} \quad \forall\ i,j \neq 0,0 \tag{67}$$

where $C^{\Delta}_{i,j}$ is the i,j coefficient of the $\Delta Block$, $C^{\Sigma}_{i,j}$ is the i,j coefficient of the $\Sigma Block$, $C^{0}_{i,j}$ is the i,j coefficient of the current block and $C^{1}_{i,j}$ is the i,j coefficient of the previous block. Notice that the DC coefficient is not considered. Once both predicted blocks are calculated, the total bit cost of the AC coefficients of the current block, the $\Delta Block$ and the $\Sigma Block$ are calculated. The one which requires less bits is chosen and transmitted. Since there are 3 possibilities for the coding (**No prediction**, **Δ** or **Σ**), an indicator must be transmitted to the decoder. Instead of spending 2 bits, two flags are used. The first flag indicates whether prediction is applied, differentiating between **No Prediction** and the other two coding modes. It is called *fq_prediction_on*, and it is "1" if the previous block is used for prediction, either adding or subtracting. The second flag, called *fq_prediction_sign*, is used in the case that *fq_prediction_on* is "1". It indicates which of the remaining two prediction modes is used. If the residual block of the prediction is obtained by subtracting the previous block, this flag is set to "1", in the case addition is used, its value becomes "0". These modes are displayed in table 13, where the correspondence bit codes for each prediction mode are depicted. This way one bit is saved every time no prediction is used.

Mode	fq_prediction_on	fq_prediction_sign
No Prediction	0	
Δ	1	1
Σ	1	0

Table 13: Frecuency Prediction modes bit code words

In a more general case that the predictor block and the one to be predicted have different zero-valued sub-band patterns, the process is the similar, but acting only on those coefficients that are not already considered zero in the pattern of the current block. In Figure 41 there is an example, and a more extended block diagram compared to Figure 39. On the other hand, Equation 66 becomes Equation 68, where $P_{i,j}$ is the position i,j of the current zero-valued sub-bands pattern. In this example the current block is **NOSYMCOL3**, and the predictor is **NOSYMROW2**. So the subtraction/addition is performed on all coefficients except those of column 3.

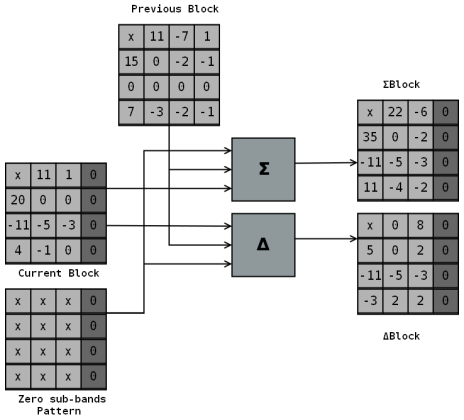


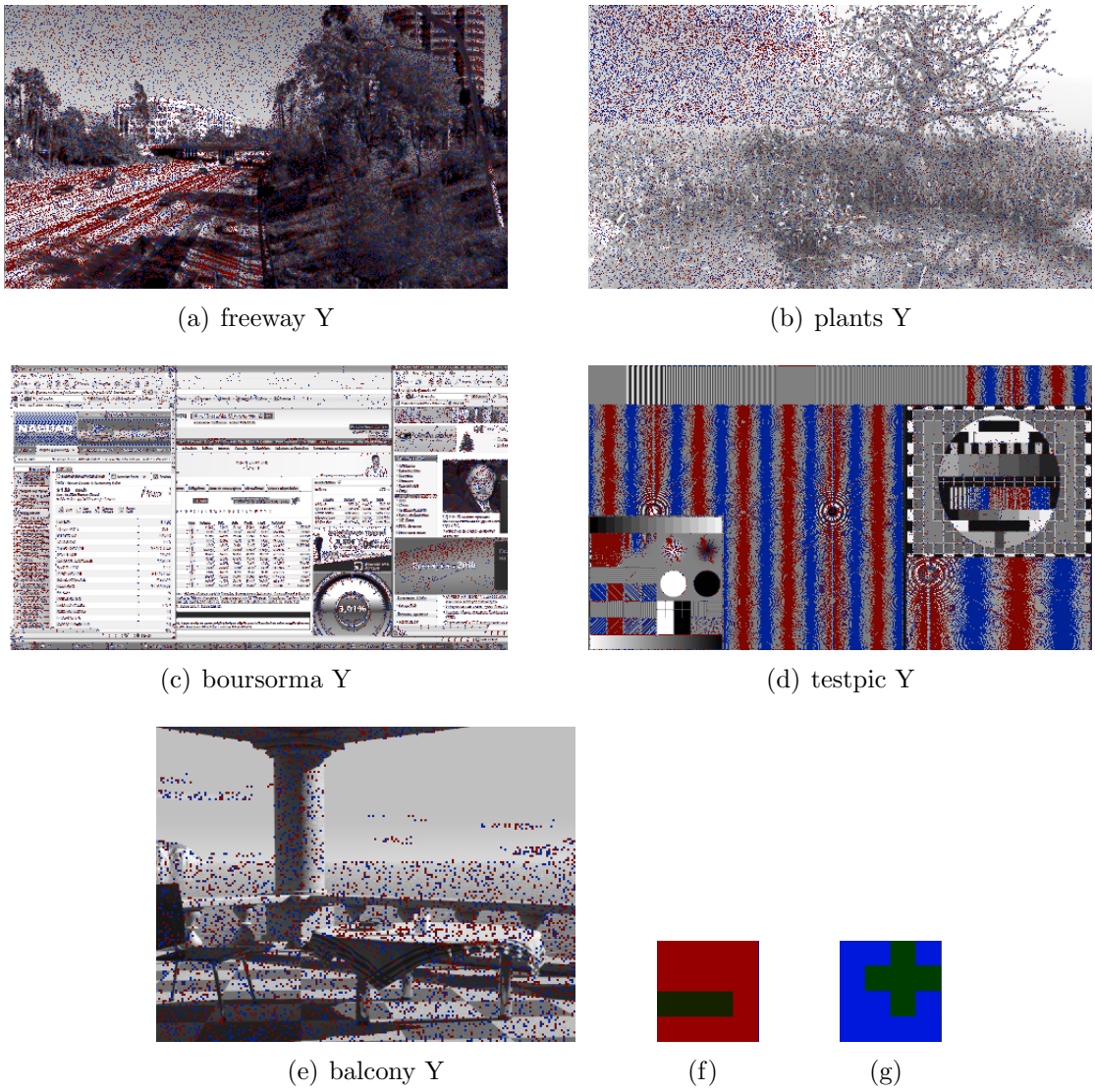
Figure 41: Different zero-valued sub-band patterns for current and predictor block

$$C^{\Delta}_{i,j} = C^{0}_{i,j} - C^{1}_{i,j} \quad if P_{i,j} \neq 0 \tag{68}$$

$$C^{\Sigma}_{i,j} = C^{0}_{i,j} + C^{1}_{i,j} \quad if P_{i,j} \neq 0 \tag{69}$$

A special situation occurs when two predicted blocks result in the same bit cost for the AC coefficients, i.e. the $\Delta Block$ and $\Sigma Block$ result in the same bit cost, which is lower than the original block (**No Prediction mode**). In that case the prediction mode which is first placed in Table 13 is chosen.

In Figure 42 the spatial distribution of the different frequency prediction modes is depicted using the colors red and blue. Notice that non-colored pixels correspond to those blocks where no frequency prediction is used, either because the bit cost of not doing it was cheaper than doing it, or because they are coded using an even more efficient coding method. Based on observation we can conclude that in those flat zones where spatial prediction already does a good performance, no prediction in frequency domain is made.



(a) freeway Y (b) plants Y

(c) boursorma Y (d) testpic Y

(e) balcony Y (f) (g)

Figure 42: Distribution of frequency prediction modes in test images, where (f) represents those blocks where Δ mode has been selected and (g), those blocks with Σ mode.

5.4.3 2D Frequency Prediction: Prediction modes extension

Similar to the prediction mode discusse in subsubsection 5.4.2, where DCT coefficients of a DCT block are predicted from the previous DCT block, it is possible to extend the number of prediction modes, using more neighboring blocks. For example the current block can be predicted using the block above or the block diagonal above of the current block, as shown in Figure 43.

For the situation that 5 modes are used see Table 14, a different coding technique for coding the mode is utilized. The *fq_prediction_on* flag is still used, working in the same way. For the situation *fq_prediction_on* is activated, then the frequency

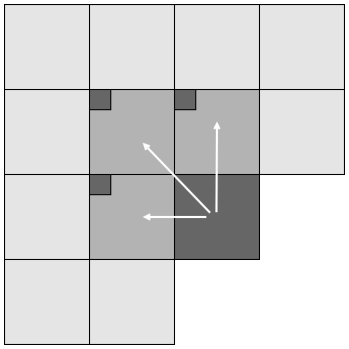


Figure 43: Blocks utilized for prediction: the darkest is the current block, and arrows indicate the predictors

prediction mode has to be send with 2 bits since it can take 4 different values: $\Delta Block$,$\Sigma Block$,$\Delta UpBlock$ and $\Sigma UpBlock$. In Table 14 the respective codewords are written. An experimental setup has been realized exploring the 5 different modes. From this experimental setup it can be concluded that the gain is very limited, and the bit costs overhead required to code more modes eliminates the coding gain. Even for the situation that extending to 7 coding modes by considering also the diagonal up block does not result in a significant coding gain, which makes these additional coding modes obsolete.

Mode	Position	Codeword
No Prediction	0	-
$\Delta LeftBlock$	1	00
$\Sigma LeftBlock$	2	01
$\Delta UpBlock$	3	10
$\Sigma UpBlock$	4	11

Table 14: Frequency Prediction: using 5 modes

5.5 Maximum Dynamic Range-Valued Patterns (MRD-V Patterns)

Blocks using the NOSYM(18) coding pattern mode are the most expensive in terms of bit cost, since it is needed to transmit all the 15 AC coefficients of the block. However, observation revealed that many times it is only one ore two coefficients in the block, which require a higher dynamic range (DR) when compared to the other sub-bands and therefore wasting bits in those coefficients which would require less bits. To battle this phenomenon a dedicated pattern is developed indicating the location of high dynamic range sub-band(s).

After Frequency Prediction, the DR of the maximum amplitude of the energy coefficient is calculated. The DR determines the amount of bits needed to code each AC coefficient. The objective of these patterns is to locate all the coefficients in the block that require this DR.

Each NOSYM block is considered as two sub-blocks, which is split over the 4×4 sized block, whereby the 4×4 block is divided by the anti-diagonal following a boundary as depicted in Figure 44. On each part, a local DR is calculated. Being $ACDR^{Up}$ the DR needed to code the maximum amplitude in the upper part of the block, and $ACDR_{Down}$ the DR to code the maximum amplitude in the lower part. This results in a marginal improvement, since the most of the times high frequency coefficients located in the lower part of the block present lower amplitudes when compared to the low frequency coefficients in the upper part. Taking advantage of the two

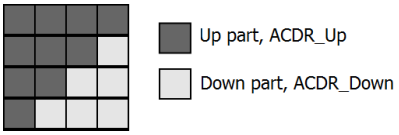
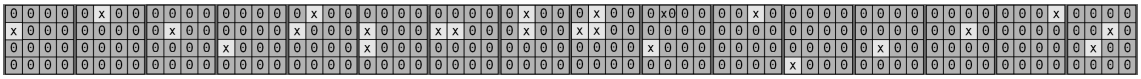


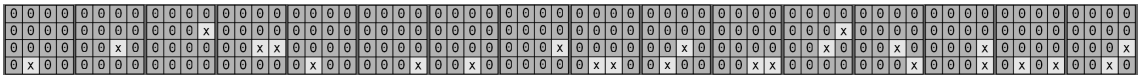
Figure 44: Up and Down parts of a NOSYM block

different DR's that are present in a NOSYM block, the pattern will distinguish between those blocks where $ACDR^{Up}$ is greater, lower, or equal to $ACDR_{Down}$. For each situation a different buffer, $Buffer_{Up}$,$Buffer_{Down}$,$Buffer_{Equal}$ respectively, with the last 16 patterns that have occurred will be updated. Each position in the buffer is numerated from 0 to 15. For the first block of each plane, these buffers are initialized with a set of patterns previously calculated based on the study of the most common occurrences in the set of test images. These patterns are shown in Figure 45. The positions marked with an "X" are energy coefficients that need $ACDR^{Up}$ or $ACDR_{Down}$ bits to be coded depending on their position, the ones with "0" are the DC coefficient and those which require a lower DR. The all "0" pattern is special case that will be explained later.

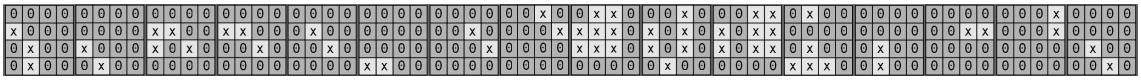
It is possible that none of the patterns on the correspondent buffer in a certain moment matches with the DCT block, but if any does, it can be only one. To select a certain pattern for a block, on one hand, all the "X" positions of the pattern must be coefficients that require $ACDR^{Up}$ if $ACDR^{Up}$ is greater than $ACDR_{Down}$,



(a) $Buffer_{Up}$



(b) $Buffer_{Down}$



(c) $Buffer_{Equal}$

(d) All zeros pattern

Figure 45: Original Patterns Buffers

$ACDR_{Down}$ if $ACDR^{Up}$ is lower that $ACDR_{Down}$, or any of them if they are equal. On the other hand, those positions with "0" in the pattern must have a lower DR. The algorithm checks through the correspondent buffer, except for the pattern with all zeros, and when it finds the one that matches with the DCT block it stops and selects that pattern. If it does not find any, it selects the one with all zeros. In Figure 46 there is an example of pattern match with a DCT block.

DCT Block

x	8	0	-1
-15	0	2	1
3	5	-2	1
-3	2	0	1

Pattern

0	X	0	0
X	0	0	0
0	0	0	0
0	0	0	0

Figure 46: Pattern Match
$ACDR^{Up} = 4$ bits
$ACDR_{Down} = 2$ bits
AC coefficients with DR = $ACDR^{Up} \Rightarrow C_{1,0}, C_{0,1}$

The current DCT block of the example has $ACDR^{Up} = 4$ and $ACDR_{Down} = 2$, so the algorithm calculates the pattern of the block pointing those coefficients of the upper part which requires $ACDR^{Up}$ bits. Next it searches for a pattern that fits with the one calculated in the $Buffer_{Up}$ since $ACDR^{Up} > ACDR_{Down}$. If we assume that for this example the state of the buffer is as shown in 48(a), the pattern calculated is on position 4. Hence, the parameter *pattern_mode* that will be sent to the decoder is set to 4. From now on, the term $ACDR$ refers to the highest DR of the 4×4 block, which can be either the $ACDR^{Up}$ or $ACDR_{Down}$ and determines the part of the block to work on. In this example, $ACDR = ACDR^{Up} = 4$. Next step is to decrease the amplitude of the selected coefficients, which are indicated by the selected pattern. The absolute value of each selected coefficient is subtracted from the value 2^{ACDR-1}. This concept avoids transmission of the first bit of the coefficients since we know it will be always "1" for the "X" positions of the pattern

and “0”, for the “0” positions. See Table 15 for more explanation. After this process all the coefficients of the upper part need one bit less for coding. Figure 47 shows the final block. Notice that for the situation in which the absolute value of one of the selected coefficients equals 2^{ACDR-1} as in the case of $C_{1,0} = 8 = 2^{ACDR-1} = 2^3$ the new value is zero, requiring no sign flag. However it is necessary to send the sign bit since for the situation of negative coefficients, the addition of 2^{ACDR-1} in the decoder results in a positive coefficient when originally it was negative. It could be avoided subtracting $2^{ACDR-1} - 1$ instead of 2^{ACDR-1} and in this way guarantee the resulting coefficient can not be 0 and so it preserve its sign. But this can cause problems in the sense that this way it is not guaranteed that all the coefficients will require 1 bit less. This occurs for the those coefficients which amplitude is equal to $2^{ACDR-1} - 1$. This is the case of the $C_{1,0} = -15$ in the example. After the subtraction of $2^{ACDR-1} - 1 = 2^3 - 1 = 7$ it would became equal to 8, which still requires 4 bits to be coded.

Coefficient	Sign	Amplitude		Subtracted	Sign	Amplitude
		1^{st}bit	Rest			
8	0	1	000	0	0	000
0		0	000			
-1	1	0	001			
-15	1	1	111	-7	1	111
2	0	0	010			
3	0	0	011			
5	0	0	101			
-2	1	0	010			
-3	1	0	011			

Table 15: Binary interpretation of subtraction of $2^{ACDR^{Up}-1}$

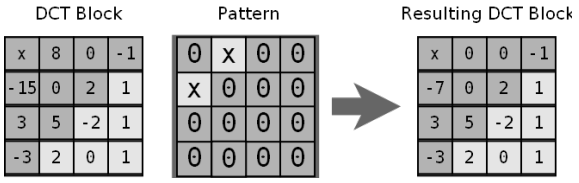
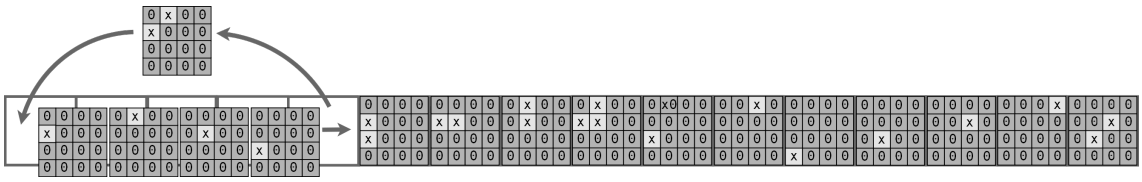


Figure 47: Resulting Block

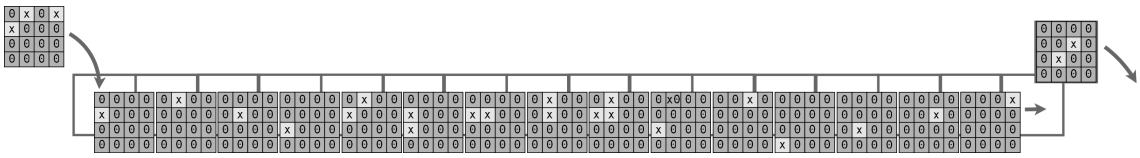
For the situation that the pattern correspondent to the DCT block is not present in the correspondent buffer, the "'All"' zeros pattern is selected by activating the *pattern_all_zeros* flag. This way the block remains the same because the decoder will not have the requiring pattern in its buffer and so it would not be able to restore the original coefficients.

After the selection of the appropriate pattern for the current DCT block and the convenient modifications on it, the correspondent buffer is updated. In the case that *pattern_all_zeros* is activated and the required pattern was not present in the

needed moment, it is added in the first position of the correspondent buffer. The pattern in the last position is discarded and all the rest are shift one position to the right. This way, considering that the next blocks will present certain correlation with the actual one, and maybe present the same distribution on its coefficients, the pattern will be available for them. Also, putting it on the first position implies that if one of the next blocks requires it, it will take less time to arrive to its position inside the buffer. On the other hand, if *pattern_all_zeros* it is not selected, and the pattern needed is already in the buffer and therefore selected, it is moved to the first position, and all the patterns that were occupying a prior position are shifted one position to the right. In this case no pattern is discarded. In Figure 48 this process is depicted.



(a) Patter found.



(b) No pattern found.

Figure 48: Buffer of patterns update.

Concluding, for the coding of the used pattern, if the correspondent pattern for the current DCT block is present in the correspondent buffer, *pattern_all_zeros* is set to "0" and *pattern_mode* takes the position of the pattern in the buffer. As the buffer has 16 position, 4 bits are needed to code *pattern_mode*, resulting in a total of 5 bits. If the required pattern is not in the buffer, it is only necessary to send *pattern_all_zeros* set to "1". It is not necessary to send which buffer is being used, because the decoder is able to decide it comparing both $ACDR^{Up}$ and $ACDR_{Down}$. Hence, the gain will depend on which part of the block is modified. In the case of the upper part, it contains 9 AC coefficients, and the lower part 6 AC coefficients. In the case of $ACDR^{Up} > ACDR_{Down}$, that is the most of the times, the 9 coefficients are coded with one bit less than before, so the gain would be $9 - 5 = 4$. If $ACDR^{Up} < ACDR_{Down}$,that is the least of the times, it would be $6 - 5 = 1$, and finally if $ACDR^{Up} = ACDR_{Down}$, the gain would be $9 + 6 - 5 = 10$. All these values are in the case that all the coefficients are coded without using any fixed codes. If it is done, gain can be different, both greater or lower depending on the coefficients. The method for coding the block coefficients will be discussed in the following section.

5.6 Coding & Syntax

This sections is a summary of how all the syntax needed by the decoder is sent by the encoder. Some points have been already explained along the previous sections, but some of them are introduced now, gathering all the syntax format of this CODEC. For the final DCT coefficients no Entropy Coding (EC) or Variable Lengh Coding (VLC) are used since they require more computation. Since it is needed to know how many bits each coefficient consumes so the decoder can read it from the stream, Dynamic Range Coding (DRC) is used instead.

5.6.1 Zero-valued sub-band Pattern Mode

The coding of the Zero sub-band pattern mode is done by using two flags and the number value of the coding pattern if its required. Flags are *mpp_symmetry_mode* and *high_symmetry_modes*. *mpp_symmetry_mode* is first sent. It is set to "1" if mpp equals to the selected mode. In case it does not, it is set to "0" and *high_symmetry_modes* is sent. If *high_symmetry_modes* is "0" then the mode belongs to the first part of Table 12 in subsection 5.3. As the zero sub-band coding mode is between UNISAD(0) and NOSYMROW3COL2(10) it is sent using 4 bits. If *high_symmetry_modes* is "1", then the zero sub-band coding mode is between SYMSAD(11) and NOSYM(18), it is subtracted the value 11 and then it becomes from 0 to 7, and is sent using 3 bits.

5.6.2 Spatial Prediction Mode

For the spatial prediction mode is always first sent the flag *most_probable_mode*. If the most probable mode, which can be calculated in the decoder without extra information, equals with the actual one, then *most_probable_mode* is set to "1", if not "0". In negative case, if the actual spatial prediction mode is greater than the most probable mode, it is decreased one unity. This way the range of the spatial prediction mode is between 0 and 7, since most probable mode is discarded, and it can be coded using only 3 bits. In the decoder, in case that *most_probable_mode* is "0" and the spatial prediction mode is greater or equal to the most probable one, it is increased one unity to restore the original value.

5.6.3 Frequency Prediction Mode

Frequency prediction mode is sent using two flags. The first one is *fq_prediction_on* which is "1" if prediction in frequency domain is done, and "0" if not. If *fq_prediction_on* is "1", then the second flag *fq_prediction_sign* is sent, indicating that $\Delta Block$ is used if it is "1", or $\Sigma Block$ is used if it is "0".

5.6.4 Maximum Dynamic Range-Valued Pattern

For those NOSYM(18) blocks, always that at least one of $\boldsymbol{ACDR^{Up}}$ or $\boldsymbol{ACDR_{Down}}$ is different from 0, the MDR-V pattern is sent. It is done by sending first the flag *pattern_all_zeros*. If it is "1", then no MDR-V pattern for this block was selected. If it is "0", then the pattern mode is sent using 4 bits, since there are 16 slots in each pattern buffers (subsection 5.5).

5.6.5 DC Coefficient & DC Dynamic Range

The DC coefficient is predicted from the DC coefficient of the previous block. This process is done on one side for NOSYM(18) blocks and on the other for all the others blocks. The Dynamic Range (DR) of the resultant amplitude is sent to the decoder. But in order to take advantage of its correlation the DR is delta coded. This means that the actual DR needed is subtracted from the previous one (notice that the same distinction between NOSYM(18) blocks and the others is also done here, meaning that DR delta is done bewteen NOSYM(18) blocks on one side, and between all the others on the other side). This difference is called **DCdeltaDR**. It is Rice coded, then sign bit is added at the end if **DCdeltaDR** is not 0 and finally it is sent.I.e., if **DCdeltaDR** is -2, then the codeword would be $\underbrace{11}_{\text{amplitude}}\ \underbrace{0}_{\text{end}}\ \underbrace{1}_{\text{sign}}$.
Then DC' amplitude is sent using $DR_{DC} - 1$ avoiding the first bit which will be always "1"(subsubsection 5.4.1).

5.6.6 AC Coefficients & AC Dynamic Range

The coding of these parameters is done in different ways on the blocks dependent on its Zero sub-bands coding pattern. The easiest case is for UNIFORM(0) blocks. In this case, no AC information is required since the block contains only the DC coefficient.
In the case that the actual block Zero sub-band coding pattern is not UNIFORM(0) nor NOSYM(18), the zero-valued sub-bands coefficient located in the coding pattern are not considered for sending. Then the DR of the maximum value of the block is calculated. This parameter is called $ACDR$. As it is done with DR_{DC}, $ACDR$ is delta and Rice coded. The previous $ACDR$ used to do the difference and calculate **ACdeltaDR** is the previous not UNIFORM(0) nor NOSYM(18) block. After the Rice coding, the sign bit is added if **ACdeltaDR** is not zero. If $ACDR$ is equal to 0, then no more AC data is sent. This means that after all the decorrelation processes (i.e, spatial and frequency prediction, MDR-V patterns,...) the block has been reduced to zeros. Otherwise, the 5 lowest possible values, $0, \pm 1, \pm 2$ are fixed coded using two bits plus the sign bit for the non zero cases. All the other coefficients which amplitude is higher than 2, are coded adding a 2 bits prefix to scape the fixed coding, then the amplitude minus 2, and finally the sign bit. See Table 16, Table 17, Table 18 and Table 19 for better understanding.

Coefficient	Codeword
0	00
±1	01S
±2	10S
skip code	11

Table 16: Fixed coded values and codewords.

DCT Block	Coefficient	Prefix	Amplitude	Sign
$\begin{bmatrix} X & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ HORIZUNISAD(1)	1	-	1	0
	0	-	0	-
	-1	-	1	1
	$ACDR = log_2(1+1) = 1$			

Table 17: Example (a).

Note that calculation of $ACDR$ has some singularities. If the maximum AC coefficient is higher than 2 it means that there will be some coefficients fixed coded and others coded using $ACDR$. But in order to decrease its value, when the maximum coefficient in the block is higher than 3, 2 is subtracted (notice that this is done for amplitudes greater than 3 because subtracting 2 from 3 would result in 1 which is a special coding case that will be explained in next lines). Notice that this way $ACDR$ may decrease and it will be no confusion with fixed coefficients since the prefixed is used. Lets go through the examples for a better explanation. In $Example(a)$ of Table 17 we have a HORIZUNISAD(1) DCT block which maximum amplitude is 1. $ACDR = \lceil log_2(1+1) = 1 \rfloor$. This is a special case, previously commented. When $ACDR$ equals to 1, just DR coding is done. So no prefixes are used, just 1 bit for the amplitude 1/0 and 1 bit for sign if amplitude is 1.
In $Example(b)$ of Table 18 the maximum coefficient is 4. As it is bigger than 3, to calculate $ACDR$ we subtract 2 to it, that will be added again in the decoder, since it will be able to distinguish the coefficient because of the prefix 11 that will be placed before the amplitude value. This way $ACDR = \lceil log_2(4-2+1) = 2 \rceil$. In this case we will have fixed and DR coded coefficients. For the zero and -1 coefficient we use the codewords of Table 16. But for the coefficient of amplitude 4, we will code $4-2=2$ which requires $ACDR = 2$ bits, plus the skip prefix 11, since originally the coefficient was greater than the maximum fixed coded value plus one that is $2+1=3$. At the end the sign bit is added. So, the codeword of the 4 coefficient is $\underbrace{11}_{skip\ prefix}\ \underbrace{10}_{amplitude}\ \underbrace{0}_{sign}$.
In $Example(c)$ of Table 19 the maximum coefficient is 3. In this case, no subtraction is done because if we do it, $ACDR = \lceil log_2(3-2+1) \rceil = 1$ and it would be confused with the coding mode when $ACDR$ is equal to 1. Since $ACDR$ determines which kind of coding will be done, and how the decoder should read the bit stream, the

DCT Block	Coefficient	Prefix	Amplitude	Sign
$\begin{bmatrix} X & 4 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ HORIZUNISAD(1)	4	11	10	0
	0	00	-	-
	-1	01	-	1
	$ACDR = log_2(4 - 2 + 1) = 2$			

Table 18: Example (b).

DCT Block	Coefficient	Prefix	Amplitude	Sign
$\begin{bmatrix} X & 3 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$ HORIZUNISAD(1)	3	11	11	0
	0	00	-	-
	-1	01	-	1
	$ACDR = log_2(3 + 1) = 2$			

Table 19: Example (c).

subtraction of 2 is only done when the maximum coefficient is greater than 3. This way, the 0 and -1 coefficient of the first line would be fixed coded following Table 16 and the 3 coefficient would be: $\underbrace{11}_{skip}\ \underbrace{11}_{prefix}\ \underbrace{11}_{amplitude}\ \underbrace{0}_{sign}$.

The last case is for NOSYM(18) blocks. For these blocks the algorithm is different. First of all because these blocks are divided in two parts, the upper part and the lower part (Figure 44 in subsection 5.5) , each one with its DR, $\boldsymbol{ACDR^{Up}}$ and $\boldsymbol{ACDR_{Down}}$ respectively. These values are delta and Rice coded in the same way than for the other zero-valued sub-band coding patterns, but in this case, $\boldsymbol{ACDR^{Up}}$ is predicted from the previous block if it is not UNIFORM(0) (from $ACDR$ if it is not NOSYM(18) or from $\boldsymbol{ACDR^{Up}}$ if it is NOSYM(18)). The $\boldsymbol{ACDR_{Down}}$ is always predicted from the $\boldsymbol{ACDR_{Down}}$ of the previous NOSYM(18) block. For these blocks there are two ways of coding the AC coefficients. Both are calculated, and then the one resulting in the lower bit cost is selected and signalized by the flag *NOSYM_coding_mode*. The firs option is to code each part the same way that was explained in the previous case, with fixed coded and DR coded coefficients dependent on its amplitude. But in this case, each part of the block, upper and lower part, use its own DR, $\boldsymbol{ACDR^{Up}}$ or $\boldsymbol{ACDR_{Down}}$, and no coefficients is subtracted 2. Splitting the block in these two parts the fact that most of the times high frequency coefficients (lower part) requires lower DR is used. For this case *NOSYM_coding_mode* is set to "0".

The second possibility is coding all the coefficients in each part with its DR, not using fixed coding. This may work when no low amplitudes are present or when all of them are low. In this case *NOSYM_coding_mode* is set to "1". In any case, if $\boldsymbol{ACDR^{Up}}$ or $\boldsymbol{ACDR_{Down}}$ is 0, no AC information of the correspondence block

part is sent, and if any of them is 1, they are always fixed coded using 1 bit plus 1 bit if amplitude is 1. In the following tables the different possible cases will be explained:

DCT Block	Coefficient	Prefix	Amplitude	Sign
$\begin{bmatrix} X & 4 & 0 & -1 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Upper part			
	4	11	100	0
	-1	01	-	1
	2	10	-	0
	0	00	-	-
	Lower part			
$ACDR_{Up} = \lceil log_2(4+1) \rceil = 3$	-2	10	-	1
$ACDR_{Down} = \lceil log_2(2+1) \rceil = 2$	1	01	-	0
$NOSYM\_coding\_mode = 0$	0	00	-	-

Table 20: Example (d).

DCT Block	Coefficient	Prefix	Amplitude	Sign
$\begin{bmatrix} X & 4 & 0 & -1 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	Upper part			
	4	-	100	0
	-1	-	001	1
	2	-	010	0
	0	-	000	-
	Lower part			
$ACDR_{Up} = \lceil log_2(4+1) \rceil = 3$	-2	-	10	1
$ACDR_{Down} = \lceil log_2(2+1) \rceil = 2$	1	-	01	0
$NOSYM\_coding\_mode = 1$	0	-	00	-

Table 21: Example (e).

In *Example (d)* in Table 20 and *Example (e)* in Table 21 it is shown the two possible modes of coding NOSYM(18) blocks depending on *NOSYM_coding_mode* flag. The decision of which one is selected depends on the bitcost. In Table 20 it is shown how for values lower than 3 they are fixed coded using a prefix, and the ones greater are coded using a skip prefix and then the amplitude value with $\boldsymbol{ACDR^{Up}}$ or $\boldsymbol{ACDR_{Down}}$ bits plus the sign bit. For Table 21, no prefix is used and the amplitudes are all coded using $\boldsymbol{ACDR^{Up}}$ or $\boldsymbol{ACDR_{Down}}$.

5.6.7 NOSYM Zero sub-band coding mode

For the blocks with NOSYM (18) zero-valued sub-band coding pattern it is possible to code the AC coefficients in two different ways as it was explained in subsubsection 5.6.7. This is decided in the bit cost function and signalized by *NOSYM_coding_mode* flag. It is set to "0" if fixed and DRC are used for the actual block, or to "1" if only DRC is used.

5.6.8 Transmition order

The order in which all the needed parameters and final coefficients are sent, if it is opportune, is the following:

1. *mpp_symmetry_mode*, *high_symmetry_modes & zero-valued_sub-band_coding_pattern*
2. *most_probable_mode* & *spatial_prediction_mode*
3. *fq_prediction_on* & *fq_prediction_sign*
4. $DCdeltaDR$
5. $ACdeltaDR$, $ACdeltaDR^{Up}$ & $ACdeltaDR_{Down}$
6. *pattern_all_zeros* & *MDR-V pattern_mode*
7. DC coefficient
8. AC coefficients

5.7 Bit Cost

There are two bit cost functions. The first one calculates only the bits needed to code the AC coefficients and it is used for deciding which Frequency prediction mode and/or MRD-V pattern will be selected. The second one is used for the bit rate control function and calculates the whole bit cost needed to code the actual block. The bit cost function for AC coefficients is done after Frequency Prediction (subsection 5.4) for not NOSYM(18) blocks, and after Frequency Prediction + MRD-V patterns for NOSYM(18) blocks. Notice that for NOSYM(18) blocks Frequency Prediction and the MRD-V pattern selection are two processes that go together and the bit cost function evaluates the result of the combination of both steps. At this point the zero-valued sub-band coding pattern is already known. The bit cost function takes into account only those coefficients that are not zero-valued sub-bands. For not NOSYM(18) blocks, the function calculates the bit cost as:

$$BC_{AC}^{SYM} = \begin{cases} 0, & \text{if } ACDR = 0; \\ N_{\text{NoZSB}} + N_{1s}, & \text{if } ACDR = 1; \\ (N_{\text{NoZSB}} - N_{fix}) \times (ACDR + 3) + (N_{fix} - N_{0s}) \times 3 + N_{0s} \times 2, & \text{otherwise} \end{cases} \tag{70}$$

where BC_{AC}^{SYM} is the AC bitcost of the not NOSYM block, N_{NoZSB} is the number of not 0 position in the Zero sub-band pattern (subsection 5.3), N_{1s}, is the number of coefficients which amplitude is equal to 1, N_0 is the number of coefficients equal to 0 and N_{fix} is the number of coefficients smaller than 3 (fixed coded). In the expression $(ACDR + 3)$ the 3 results by adding 2, meaning the two bits prefix, and 1 because of the sign bit.

For the case that the block is NOSYM(18):

$$BC_{AC}^{Up} = \begin{cases} 0, & \text{if } ACDR_{Up} = 0; \\ (9 + N_{0s}^{Up}) \times 2 + N_{0s}^{Up}, & \text{if } ACDR_{Up} = 1; \\ (9 - N_{fix}^{Up}) \times (ACDR^{Up'} + 3) + (N_{fix}^{Up} - N_{0s}^{Up}) \times 3 + N_{0s}^{Up}, & \text{otherwise} \end{cases} \tag{71}$$

$$BC_{AC}^{Dn} = \begin{cases} 0, & \text{if } ACDR_{Down} = 0; \\ (6 + N_{0s}^{Dn}) \times 2 + N_{0s}^{Dn}, & \text{if } ACDR_{Down} = 1; \\ (6 - N_{fix}^{Dn}) \times (ACDR'_{Down} + 3) + (N_{fix}^{Dn} - N_{0s}^{Dn}) \times 3 + N_{0s}, & \text{otherwise} \end{cases} \tag{72}$$

where BC_{AC}^{Up} is the AC bit cost of the upper part of the NOSYM block and BC_{AC}^{Dn} the bit cost of the lower part,the 9 and the 6 are the number of AC coefficients in each part, N_{NoZSB} is the number of not 0 position in the Zero sub-band pattern (upper or lower), N_{1s}, is the number of coefficients which amplitude is equal to 1 in the upper part or lower part, and N_{0s} is the number of coefficients equal to 0 in the upper part or lower part. In the expressions $(ACDR^{Up} + 3)$ and $(ACDR_{Down} + 3)$ the 3 comes from adding 2 bits for the prefix and the 1 because of the sign bit. Moreover, in those cases that $ACDR^{Up}$ and/or $ACDR_{Down}$ are not 0 or 1, then

$ACDR'^{Up}$ and/or $ACDR'_{Down}$ are used, where:

$$ACDR'^{Up} = \begin{cases} ACDR^{Up} - !pattern\_all\_zeros, & \text{if } ACDR^{Up} \geq ACDR_{Down} \\ ACDR^{Up}, & \text{otherwise} \end{cases} \quad (73)$$

$$ACDR'_{Down} = \begin{cases} ACDR_{Down} - !pattern\_all\_zeros, & \text{if } ACDR_{Down} \geq ACDR^{Up} \\ ACDR_{Down}, & \text{otherwise} \end{cases} \quad (74)$$

Notice that $ACDR_{Up}$ and/or $ACDR_{Down}$ may not be the DR needed to code the maximum amplitude, but one unity less if the flag *pattern_all_zeros* is desactivated, meaning a MDR-V pattern is used and so the DR decreased. Finally the total bit cost if *NOSYM_coding_mode* is set to 0, meaning fixed and DR coding is used, is:

$$BC_{AC}^{NOSYM} = BC_{AC}^{Up} + BC_{AC}^{Dn} + S_{MDRVpattern} \quad (75)$$

where $S_{MDRVpattern}$ is the amount of sign bits needed if after subtracting the MDR-V pattern some coefficient turns to 0. In the case that *NOSYM_coding_mode* is set to 1, meaning that all coefficients are coded with $ACDR_{Up}$ or $ACDR_{Down}$ bits, the final bit cost is:

$$BC_{AC}^{NOSYM} = 9 \times ACDR'_{Up} + 6 \times ACDR'_{Down} + (9 + 6 - N_{0s}^{Up} - N_{0s}^{Dn}) + S_{MDRVpattern} \quad (76)$$

5.8 Bit Rate Control

For this thesis a very fast, cheap and easy implementing bit rate control has been developed. There is no need to transmit the QP, since the decoder is able to calculate it from previous data. The disadvantage of this method is that there is no control over the image quality. It only almost guarantee a certain compression ratio per video frame line (taking into account the three channels). The only parameter in the designed video codec with which is possible to manage the bit cost of a certain block is the QP factor. Because of this fact, the bit rate control algorithm will only act over this parameter. At the beginning of each plane a defined QP_0 is used. After processing the complete block, the total bit cost including header and coefficients is calculated, BC_{Total}. At this point it is possible to calculate the compression ratio achieved in the current block as:

$$R_{Block} = \frac{4 \times 4 \times bpp}{BC_{Total}} \tag{77}$$

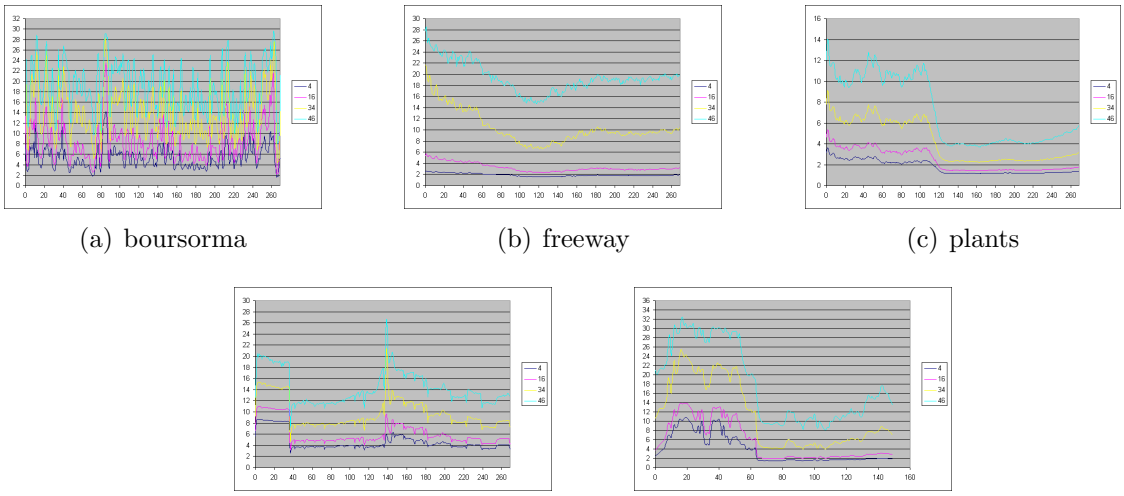
where bpp is the bit depth of the video data. Then, R_{Block} is compared with the required compression ratio $R_{Required}$. In case that R_{Block} is lower than $R_{Required}$ then for the next image block QP is increased an unity. If instead of that, R_{Block} is greater than $R_{Required}$, then QP is decreased an unity in order to get better image quality limiting quantization errors. Another limitation that has to be taken into account is that QP must be always $QP_{min} \leq QP \leq Q_{max}$ with $QP_{min} = 4$ (no compression) and $QP_{max} = 51$. In order to take advantage of the greater decorrelation in the UV planes in comparison with the Y plane, a good idea might be to define an R_{Offset} that modifies the $R_{Required}$ in each channel.For Y plane, $R_{Required}$ passes to be $R_{Required} - 2 \times R_{Offset}$ and for each UV planes, $R_{Required}$ passes to be $R_{Required} + R_{Offset}$. For this thesis R_{Offset} has been chosen as $R_{Offset} = 0.2$. Doing this, the total $R_{Required}$ is preserved, but some more bits are used for coding Y signal, where error because of quantization are more dangerous and notorius in the restored image.

From the second line on, the line QP_0 is not the one defined for the beginning of a plane. In order to attempt to arrive to a value of QP that achieves the desired $R_{Required}$ faster, the first QP of the a line l, QP_0^l is calculated as the mean QP used in the previous line:

$$QP_0^l = \frac{\sum_i^{Width/4} QP_i^{l-1}}{Width/4} \tag{78}$$

where Width/4 is the amount of blocks in a frame line since $Width$ is the image width in pixels and 4 is the block length, and the l super index indicates the line index.

In order to compare the performance of the codec when working with a fixed QP and no bitrate control, or in the other case, when working with bit rate control and a fixed compression ratio per line, the following graphs and tables are added. Figure 49 shows the variation of the compression ratio in each line with a certain QP. Table 22 shows the PSNR of the reconstructed image using a certain fixed QP. Finally, Table 23 shows the PSNR in each plane when a fixed ratio is selected.



(a) boursorma (b) freeway (c) plants

(d) testpic (e) balcony

Figure 49: Compression ratio achieved per frame line when fixed QP is selected (no bit rate control). Each color represents a different QP.

Image	QP	PSNR (dB)		
		Y	U	V
boursorma	4	65.82	66.23	66.18
	16	61.66	62.57	62.58
	34	45.41	47.76	47.85
	46	34.33	38.23	38.4
freeway	4	63.3	63.27	63.27
	16	57.41	57.66	57.46
	34	42.75	44.42	43.87
	46	34	38.44	37.97
plants	4	4	76.28	76.44
	16	16	70.66	70.74
	34	34	53.59	54.95
	46	46	42.45	46.97
testpic	4	63.76	inf	inf
	16	58.29	inf	inf
	34	41.88	inf	inf
	46	30.72	inf	inf
balcony	4	76.28	76.44	76.37
	16	70.66	70.74	70.87
	34	53.59	54.95	56.2
	46	42.45	46.97	49.02

Table 22: PSNR of reconstructed image at fixed QP

Image	Ratio	PSNR (dB)		
		Y	U	V
boursorma	2	56.46	63.2	63.64
	3	44.96	58.95	60.09
	4	41.8	55.3	57.05
	5	40.63	53.28	54.96
freeway	2	61.76	60.92	60.11
	3	52.84	55.84	54.6
	4	47.21	52.27	50.97
	5	43.92	50.42	49.18
plants	2	51.95	56.66	62.68
	3	40.02	47.47	53.55
	4	37.46	42.84	48.97
	5	37.07	40.64	46.45
testpic	2	49.34	108.05	108.05
	3	37.08	108.05	108.05
	4	31.82	108.05	108.05
	5	29.51	108.05	108.05
balcony	2	69.36	71.21	74.04
	3	54.15	63.34	67.71
	4	47.14	58.68	63.15
	5	43.57	56	60.61

Table 23: PSNR of reconstructed image at fixed compression ratio

6 Conclusions

Embedded Video Compression (EVC) in modern digital TV-systems are subjected to implementation constraints, which are more demanding than for traditional video compression. As a result, obtaining high compression performance in terms of compression ratio and picture quality with limited resources is not a trivial task. It is for this reason, that the de-correlation and quantization phase need to be highly efficient with respect to compression performance, computational effort and memory footprint.

At the same time, in digital TV-systems there is a wide variety of video characteristics such as graphics, natural video, captured from the camera or digitally generated, etc. It is for this reason that different kind of techniques may be utilized in order to fulfill the compression requirements for a broad range of video data.

From the different tests conducted in different color spaces discussed in this thesis show that YUV color space is the best color space. It shows to have a good coding performance regarding spatial prediction, when using fix QP compression. However in H.264 Fidelity range extensions (FRExt) other color domains such as RGB or YC_oC_g are also allowed, enabling lossless compression, avoiding losses due to change of color domain.

HSV color domain initially looked promising due to the bit depth reduction needed to code its components in comparison with RGB or the other luminance color spaces when working with high dynamic range images (10,12 bpp), since two of its components can be coded with less bits. I.e. the angle H component is coded with 9 bits since its dynamic range goes from 0 to 359. The saturation component S is a parameter which value is between 0 and 1, whereby its accuracy can be selected resulting in more or less bits for coding. In this thesis it has been coded using 8 and 9 bits per sample. Furthermore, for images not completely saturated it would be expected that S shows a low variance in its values. Nevertheless, in this implementation, the results of testing were not satisfactory, in the sense that bit cost achieved was worse and $PSNR$ of the restored images decayed much more faster for increased QP than in the other color spaces, due to the very high sensitivity to quantization and rounding of DCT coefficients of the H and S components. Nevertheless, HSV may be more useful in combination with a lossless coding chain.

It has been observed that in graphic data it is common to find certain structures in spatial domain that is repetitive in a certain direction and produce certain patterns in the DCT domain. This patterns correspond to symmetries in the DCT block or a clustering of zero-valued sub-bands. Efficient coding of these patterns contributes to the coding gain when many coefficients of the DCT block can be clustered and coded simultaneously. These technique works better for graphic data since its artificial structure facilitates the appearance of symmetries and zero-valued coefficients in the DCT domain. Nevertheless, for natural video it also appears to be useful since the low-pass characteristics of natural video result in complete rows, columns or combination of them with zero-valued high frequency sub-bands. Moreover, after

the quantization step, the reduction and sometimes the cancellation of certain sub-bands contribute to the presence of these zero-valued rows and columns.

Another observation was that due to the spatial correlation between pixels, in many situations, pixels in a block are very similar to the pixels at the block boundary, enabling prediction of the pixels inside the block. This prediction that is already implemented in H.264 and has good decorrelation performance for video originated from a natural video source. However, in artificial video data such as graphic data this prediction technique will fail due to the introduction of mosquito noise after quantization. An important point regarding spatial prediction is that the residual information can still benefit from the zero-valued sub-band coding concept. However, spatial prediction must be controlled when coding graphic data as it may cause mosquito noise around steep edges.

Guided by the idea of the spatial prediction, prediction in the transformed domain was investigated. The experimental results, which where obtained using only information in scanning order direction indicate that there is still some correlation between neighboring DCT blocks and its coefficients. It was observed that in many situations a DCT coefficient at a certain frequency had similar energy to the DCT coefficient at the same frequency in the following DCT block. This resulted in a concept to predict the actual DCT block from the previous DCT block. This sometimes results in an increase of energy for a specific AC coefficient, due to difference in sign. In order to avoid this amplitude increase, the $\Delta Block$ and $\Sigma Block$ predictors were constructed resulting in a better de-correlation performance. Furthermore, the energy located at the zero-valued sub-bands pattern positions are don't care, such that a change of energy due to subtraction or addition does not lead to additional energy.

Based on the promising results obtained by prediction in the frequency domain, the degree of freedom for prediction has been extended to 2D, enabling transform data from the previous slice to be used as predictor. Unfortunately, this extension did not result in an increased de-correlation performance. The bit cost increase due to the appearance of more prediction modes resulted in a very low or even negative gain. The root cause for this lies in the fact that the previous block in scanning order is the best prediction candidate and the upper blocks are rarely selected as a predictor.

After prediction in the frequency domain, the residual frequency information shows a more or less flat amplitude behavior, whereby only a few sub-bands have a significant higher amplitude. When using Dynamical Range Coding (DRC), these high amplitude coefficients are dominant. The entropy coding stage can still benefit from this when using patterns that indicate the location of the high amplitude sub-bands, clustering the coefficients that required the highest dynamic range and thus separating the low and higher dynamical range sub-bands.

6.1 Future work

It was mentioned in this thesis, that a particular color space can not always be succeeded by a transform coding kernel to de-correlate the individual color information signals. As a result of this, it is opportune to investigate the clustering properties of the different color spaces. The objective is to reveal potential clustering of equal or nearly equal values, such that these signals can be processed efficiently by other coding techniques. This may be the case for HSV color space domain.

The experimental results obtained with 2D-based prediction in the frequency domain require further investigation as the current results are counter intuitive. One would expect if more degrees of freedom are available, that this would have a positive influence on the compression performance. However, during evaluation of various coding concepts of certain coding parameters, it was observed that the gain was eliminated due to expensive decoder notification.

Clustering sub-band with a high dynamic range becomes expensive when the amount of patterns is large, when implemented in a parallel manner. It is for this reason that there is a need for a novel algorithm that efficiently determines the location of such sub-bands.

Finally, the observation of the results of the image test set confirms that the Y signal consumes the majority of bits in comparison to the chrominance signals U and V. As the CODEC works on a block by block basis, in order to be able to achieve better and fixed compression ratios, it would be useful to process the chrominance planes first in order to use the possible non used bits in the Y plane. Something similar was intended when using the parameter R_{offset} in the deigned bit rate control, but this idea should be improved.

A Appendix

A.1 Mathematical Proof of Symmetry Effects

A.1.1 DCT transform matrix

$$C = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix}, \qquad \begin{aligned} a &= 1/2 \\ b &= \sqrt{1/2}\cos(\Pi/8) = 0.6532... \\ c &= \sqrt{1/2}\cos(3\Pi/8) = 0.2406... \end{aligned}$$

A.1.2 Horizontal Symmetry

$$X = \begin{pmatrix} k & o & o & k \\ l & p & p & l \\ m & q & q & m \\ n & r & r & n \end{pmatrix}$$

$$Y = CXC^T$$

$$\begin{aligned} CX &= \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} k & o & o & k \\ l & p & p & l \\ m & q & q & m \\ n & r & r & n \end{pmatrix} \\ &= \begin{pmatrix} a(k+l+m+n) & a(o+p+q+r) & a(o+p+q+r) & a(k+l+m+n) \\ b(k-n)+c(l-m) & b(o-r)+c(p-q) & b(o-r)+c(p-q) & b(k-n)+c(l-m) \\ a(k-l-m+n) & a(o-p-q+r) & a(o-p-q+r) & a(k-l-m+n) \\ b(m-l)+c(k-n) & b(q-p)+c(o-r) & b(q-p)+c(o-r) & b(m-l)+c(k-n) \end{pmatrix} \\ &= \begin{pmatrix} x_1 & x_2 & x_2 & x_1 \\ x_3 & x_4 & x_4 & x_3 \\ x_5 & x_6 & x_6 & x_5 \\ x_7 & x_8 & x_8 & x_7 \end{pmatrix} \end{aligned}$$

$$\begin{aligned} CXC^T &= \begin{pmatrix} x_1 & x_2 & x_2 & x_1 \\ x_3 & x_4 & x_4 & x_3 \\ x_5 & x_6 & x_6 & x_5 \\ x_7 & x_8 & x_8 & x_7 \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix} \\ &= \begin{pmatrix} y_1 & b(x_1-x_1)+c(x_2-x_2) & y_3 & b(-x_2+x_2)+c(x_1-x_1) \\ y_5 & b(x_3-x_3)+c(x_4-x_4) & y_7 & b(-x_4+x_4)+c(x_3-x_3) \\ y_9 & b(x_5-x_5)+c(x_6-x_6) & y_{11} & b(-x_6+x_6)+c(x_5-x_5) \\ y_{13} & b(x_7-x_7)+c(x_8-x_8) & y_{15} & b(-x_8+x_8)+c(x_7-x_7) \end{pmatrix} \\ &= \begin{pmatrix} y_1 & 0 & y_3 & 0 \\ y_5 & 0 & y_7 & 0 \\ y_9 & 0 & y_{11} & 0 \\ y_{13} & 0 & y_{15} & 0 \end{pmatrix} \end{aligned}$$

A.1.3 Horizontal Uniformity

$$X = \begin{pmatrix} k & k & k & k \\ l & l & l & l \\ m & m & m & m \\ n & n & n & n \end{pmatrix}$$

$$Y = CXC^T$$

$$CX = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} k & k & k & k \\ l & l & l & l \\ m & m & m & m \\ n & n & n & n \end{pmatrix}$$

$$= \begin{pmatrix} a(k+l+m+n) & a(k+l+m+n) & a(k+l+m+n) & a(k+l+m+n) \\ b(k-n)+c(l-m) & b(k-n)+c(l-m) & b(k-n)+c(l-m) & b(k-n)+c(l-m) \\ a(k-l-m+n) & a(k-l-m+n) & a(k-l-m+n) & a(k-l-m+n) \\ b(m-l)+c(k-n) & b(m-l)+c(k-n) & b(m-l)+c(k-n) & b(m-l)+c(k-n) \end{pmatrix}$$

$$= \begin{pmatrix} x_1 & x_1 & x_1 & x_1 \\ x_2 & x_2 & x_2 & x_2 \\ x_3 & x_3 & x_3 & x_3 \\ x_4 & x_4 & x_4 & x_4 \end{pmatrix}$$

$$CXC^T = \begin{pmatrix} x_1 & x_1 & x_1 & x_1 \\ x_2 & x_2 & x_2 & x_2 \\ x_3 & x_3 & x_3 & x_3 \\ x_4 & x_4 & x_4 & x_4 \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix}$$

$$= \begin{pmatrix} y_1 & b(x_1-x_1)+c(x_1-x_1) & a(x_1-x_1-x_1+x_1) & b(-x_1+x_1)+c(x_1-x_1) \\ y_5 & b(x_2-x_2)+c(x_2-x_2) & a(x_2-x_2-x_2+x_2) & b(-x_2+x_2)+c(x_2-x_2) \\ y_9 & b(x_3-x_3)+c(x_3-x_3) & a(x_3-x_3-x_3+x_3) & b(-x_3+x_3)+c(x_3-x_3) \\ y_{13} & b(x_4-x_4)+c(x_4-x_4) & a(x_4-x_4-x_4+x_4) & b(-x_4+x_4)+c(x_4-x_4) \end{pmatrix}$$

$$= \begin{pmatrix} y_1 & 0 & 0 & 0 \\ y_5 & 0 & 0 & 0 \\ y_9 & 0 & 0 & 0 \\ y_{13} & 0 & 0 & 0 \end{pmatrix}$$

A.1.4 Vertical Symmetry

$$X = \begin{pmatrix} k & o & o & k \\ l & p & p & l \\ m & q & q & m \\ n & r & r & n \end{pmatrix}$$

$$Y = CXC^T$$

$$CX = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} k & o & o & k \\ l & p & p & l \\ m & q & q & m \\ n & r & r & n \end{pmatrix}$$

$$= \begin{pmatrix} a(2k+2o) & a(2l+2p) & a(2m+2q) & a(2n+2r) \\ 0 & 0 & 0 & 0 \\ a(2k+2o) & a(2l+2p) & a(2m+2q) & a(2n+2r) \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ 0 & 0 & 0 & 0 \\ x_5 & x_6 & x_7 & x_8 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$CXC^T = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ 0 & 0 & 0 & 0 \\ x_5 & x_6 & x_7 & x_8 \\ 0 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix}$$

$$= \begin{pmatrix} y_1 & y_2 & y_3 & y_4 \\ 0 & 0 & 0 & 0 \\ y_9 & y_{10} & y_{11} & y_{12} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

A.1.5 Vertical Uniformity

$$X = \begin{pmatrix} k & l & m & n \\ k & l & m & n \\ k & l & m & n \\ k & l & m & n \end{pmatrix}$$

$$Y = CXC^T$$

$$CX = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} k & l & m & n \\ k & l & m & n \\ k & l & m & n \\ k & l & m & n \end{pmatrix}$$

$$= \begin{pmatrix} 4ak & 4al & 4am & 4an \\ b(k-k)+c(k-k) & b(l-l)+c(l-l &)b(m-m)+c(m-m) & b(n-n)+c(n-n) \\ a(2k-2k) & a(2l-2l) & a(2m-2m) & a(2n-2n) \\ b(k-k)+c(k-k) & b(l-l)+c(l-l &)b(m-m)+c(m-m) & b(n-n)+c(n-n) \end{pmatrix}$$

$$= \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$CXC^T = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix}$$

$$= \begin{pmatrix} y_1 & y_2 & y_3 & y_4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

A.1.6 Diagonal Symmetry

$$X = \begin{pmatrix} k & l & m & n \\ l & o & p & q \\ m & p & r & s \\ n & q & s & y \end{pmatrix}$$

$$Y = CXC^T$$

$$CX = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} k & l & m & n \\ l & o & p & q \\ m & p & r & s \\ n & q & s & y \end{pmatrix}$$

$$= \begin{pmatrix} a(k+l+m+n) & a(l+o+p+q) & a(m+p+r+s) & a(n+q+s+t) \\ b(k-n)+c(l-m) & b(l-q)+c(o-p) & b(m-s)+c(p-r) & b(n-t)+c(q-s) \\ a(k-l-m+n) & a(l-o-p+q) & a(m-p-r+s) & a(n-q-s+t) \\ b(m-l)+c(k-n) & b(p-o)+c(l-q) & b(r-p)+c(m-s) & b(s-q)+c(n-t) \end{pmatrix}$$

$$= \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix}$$

$$CXC^T = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix}$$

$$= \begin{pmatrix} y_1 & y_2 & y_3 & y_4 \\ y_5 & y_6 & y_7 & y_8 \\ y_9 & y_{10} & y_{11} & y_{12} \\ y_{13} & y_{14} & y_{15} & y_{16} \end{pmatrix}$$

$$
\begin{aligned}
y_2 &= ab(k+l+m+n) + ac(l+o+p+q) - ac(m+p+r+s) - ab(n+q+s+t) \\
&= ab(k+l+m-q-s-t) + ac(l+o+q-m-r-s) \\
y_5 &= ab(k-n) + ac(l-m) + ab(l-q) + ac(o-p) + ab(m-s) + ac(p-r) + ab(n-t) + ac(q-s) \\
&= ab(k+l+m-q-s-t) + ac(l+o+q-m-r-s) \\
&= y_2
\end{aligned}
$$

$$
\begin{aligned}
y_3 &= a^2(k+l+m+n) - a^2(l+o+p+q) - a^2(m+p+r+s) + a^2(n+q+s+t) \\
&= a^2(k+2n-o-2p-r+t) \\
y_9 &= a^2(k-l-m+n) + a^2(l-o-p+q) + a^2(m-p-r+s) + a^2(n-q-s+t) \\
&= a^2(k+2n-o-2p-r+t) \\
&= y_3
\end{aligned}
$$

$$
\begin{aligned}
y_4 &= ac(k+l+m+n) - ab(l+o+p+q) + ab(m+p+r+s) - ac(n+q+s+t) \\
&= ab(m+r+s-l-o-q) + ac(k+l+m-q-s-t) \\
y_{13} &= ab(m-l) + ac(k-n) + ab(p-o) + ac(l-q) + ab(r-p) + ac(m-s) + ab(s-q) + ac(n-t) \\
&= ab(m+r+s-l-o-q) + ac(k+l+m-q-s-t) \\
&= y_4
\end{aligned}
$$

$$
\begin{aligned}
y_7 &= ab(k-n) + ac(l-m) - ab(l-q) - ac(o-p) - ab(m-s) - ac(p-r) + ab(n-t) + ac(q-s) \\
&= ab(k+q+s-l-m-t) + ac(l+q+r-m-o-s) \\
y_{10} &= ab(k-l-m+n) + ac(l-o-p+q) - ac(m-p-r+s) - ab(n-q-s+t) \\
&= ab(k+q+s-l-m-t) + ac(l+q+r-m-o-s) \\
&= y_7
\end{aligned}
$$

$$
\begin{aligned}
y_8 &= bc(k-n) + c^2(l-m) - b2(l-q) - bc(o-p) + b2(m-s) + bc(p-r) - bc(n-t) - c^2(q-s) \\
&= bc(2p-2n+k+t-o-r) + b2(m+q-l-s) + c^2(l+s-m-q) \\
y_{14} &= b2(m-l) + bc(k-n) + bc(p-o) + c^2(l-q) - bc(r-p) - c^2(m-s) - b2(s-q) - bc(n-t) \\
&= bc(2p-2n+k+t-o-r) + b2(m+q-l-s) + c^2(l+s-m-q) \\
&= y_8
\end{aligned}
$$

$$
\begin{aligned}
y_{12} &= ac(k-l-m+n) - ab(l-o-p+q) + ab(m-p-r+s) - ac(n-q-s+t) \\
&= ab(m+o+s-l-q-r) + ac(k+q+s-l-m-t) \\
y_{15} &= ab(m-l) + ac(k-n) - ab(p-o) - ac(l-q) - ab(r-p) - ac(m-s) + ab(s-q) + ac(n-t) \\
&= ab(m+o+s-l-q-r) + ac(k+q+s-l-m-t) \\
&= y_{12}
\end{aligned} \tag{79}
$$

A.1.7 Antidiagonal Symmetry

$$X = \begin{pmatrix} k & o & r & t \\ l & p & s & r \\ m & q & p & o \\ n & m & l & k \end{pmatrix}$$

$$Y = CXC^T$$

$$CX = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} k & o & r & t \\ l & p & s & r \\ m & q & p & o \\ n & m & l & k \end{pmatrix}$$

$$= \begin{pmatrix} a(k+l+m+n) & a(o+p+q+m) & a(r+s+p+l) & a(k+o+r+t) \\ b(k-n)+c(l-m) & b(o-m)+c(p-q) & b(r-l)+c(s-p) & b(t-k)+c(r-o) \\ a(k-l-m+n) & a(o-p-q+m) & a(r-s-p+l) & a(t-r-o+k) \\ b(m-l)+c(k-n) & b(q-p)+c(o-m) & b(p-s)+c(r-l) & b(o-r)+c(t-k) \end{pmatrix}$$

$$= \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix}$$

$$CXC^T = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix}$$

$$= \begin{pmatrix} y_1 & y_2 & y_3 & y_4 \\ y_5 & y_6 & y_7 & y_8 \\ y_9 & y_{10} & y_{11} & y_{12} \\ y_{13} & y_{14} & y_{15} & y_{16} \end{pmatrix}$$

$$
\begin{aligned}
y_2 &= ab(k+l+m+n) + ac(o+p+q+m) - ac(r+s+p+l) - ab(k+o+r+t) \\
&= ab(l+m+n-o-r-t) + ac(m+o+q-l-r-s) \\
y_5 &= ab(k-n) + ac(l-m) + ab(o-m) + ac(p-q) + ab(r-l) + ac(s-p) + ab(t-k) + ac(r-o) \\
&= ab(l+m+n-o-r-t) - ac(m+o+q-l-s-r) \\
&= -y_2
\end{aligned}
$$

$$
\begin{aligned}
y_3 &= a^2(k+l+m+n) - a^2(o+p+q+m) - a^2(r+s+p+l) + a^2(k+o+r+t) \\
&= a^2(2k+n+t-2p-q-s) \\
y_9 &= a^2(k-l-m+n) + a^2(o-p-q+m) + a^2(r-s-p+l) + a^2(t-r-o+k) \\
&= a^2(2k+n+t-2p-q-s) \\
&= y_3
\end{aligned}
$$

$$
\begin{aligned}
y_4 &= ac(k+l+m+n) - ab(o+p+q+m) + ab(r+s+p+l) - ac(k+o+r+t) \\
&= ab(l+r+s-m-o-q) + ac(l+m+n-o-r-t) \\
y_{13} &= ab(m-l) + ac(k-n) + ab(q-p) + ac(o-m) + ab(p-s) + ac(r-l) + ab(o-r) + ac(t-k) \\
&= -ab(l+s+r-m-o-q) - ac(l+m+n-o-r-t) \\
&= -y_4
\end{aligned}
$$

$$
\begin{aligned}
y_7 &= ab(k-n) + ac(l-m) - ab(o-m) - ac(p-q) - ab(r-l) - ac(s-p) + ab(t-k) + ac(r-o) \\
&= ab(l+m+t-n-o-r) + ac(l+q+r-m-o-s) \\
y_{10} &= ab(k-l-m+n) + ac(o-p-q+m) - ac(r-s-p+l) - ab(t-r-o+k) \\
&= -ab(l+m+t-n-o-r) - ac(l+q+r-m-o-s) \\
&= -y_7
\end{aligned}
$$

$$
\begin{aligned}
y_8 &= bc(k-n) + c^2(l-m) - b2(o-m) - bc(p-q) + b2(r-l) + bc(s-p) - bc(t-k) - c^2(r-o) \\
&= bc(2k+q+s-n-2p-t) + b2(m+r-l-o) + c^2(l+o-m-r) \\
y_{14} &= b2(m-l) + bc(k-n) + bc(q-p) + c^2(o-m) - bc(p-s) - c^2(r-l) - b2(o-r) - bc(t-k) \\
&= bc(2k+q+s-n-2p-t) + b2(m+r-l-o) + c^2(l+o-m-r) \\
&= y_8
\end{aligned}
$$

$$
\begin{aligned}
y_{12} &= ac(k-l-m+n) - ab(o-p-q+m) + ab(r-s-p+l) - ac(t-r-o+k) \\
&= ab(l+q+r-m-o-s) + ac(n+o+r-l-m-t) \\
y_{15} &= ab(m-l) + ac(k-n) - ab(q-p) - ac(o-m) - ab(p-s) - ac(r-l) + ab(o-r) + ac(t-k) \\
&= -ab(l+q+r-m-o-s) - ac(n+o+r-l-m-t) \\
&= -y_{12}
\end{aligned} \tag{80}
$$

A.1.8 Mirror Symmetry

$$X = \begin{pmatrix} k & o & r & n \\ l & p & q & m \\ m & q & p & l \\ n & r & o & k \end{pmatrix}$$

$$Y = CXC^T$$

$$CX = \begin{pmatrix} a & a & a & a \\ b & c & -c & -b \\ a & -a & -a & a \\ c & -b & b & -c \end{pmatrix} \times \begin{pmatrix} k & o & r & n \\ l & p & q & m \\ m & q & p & l \\ n & r & o & k \end{pmatrix}$$

$$= \begin{pmatrix} a(k+l+m+n) & a(o+p+q+r) & a(o+p+q+r) & a(k+l+m+n) \\ b(k-n)+c(l-m) & b(o-r)+c(p-q) & -b(o-r)-c(p-q) & -b(k-n)-c(l-m) \\ a(k-l-m+n) & a(o-p-q+r) & a(o-p-q+r) & a(k-l-m+n) \\ b(m-l)+c(k-n) & b(q-p)+c(o-r) & -b(q-p)-c(o-r) & -b(m-l)-c(k-n) \end{pmatrix}$$

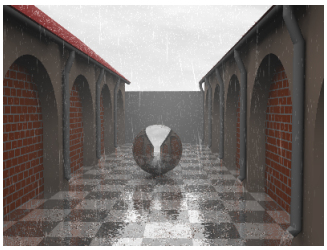
$$= \begin{pmatrix} x_1 & x_2 & x_2 & x_1 \\ x_3 & x_4 & -x_4 & -x_3 \\ x_5 & x_6 & -x_6 & -x_5 \\ x_7 & x_8 & -x_8 & -x_7 \end{pmatrix}$$

$$CXC^T = \begin{pmatrix} x_1 & x_2 & x_2 & x_1 \\ x_3 & x_4 & -x_4 & -x_3 \\ x_5 & x_6 & -x_6 & -x_5 \\ x_7 & x_8 & -x_8 & -x_7 \end{pmatrix} \times \begin{pmatrix} a & b & a & c \\ a & c & -a & -b \\ a & -c & -a & b \\ a & -b & a & -c \end{pmatrix}$$

$$= \begin{pmatrix} y_1 & 0 & y_3 & 0 \\ 0 & y_6 & 0 & y_8 \\ y_9 & 0 & y_{11} & 0 \\ 0 & y_{14} & 0 & y_{16} \end{pmatrix}$$

A.2 Complete set of test images

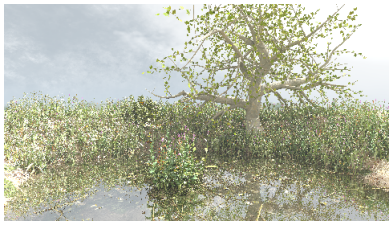
(a)



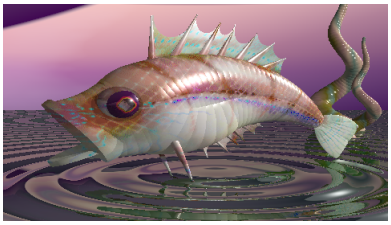
(b)

(c)

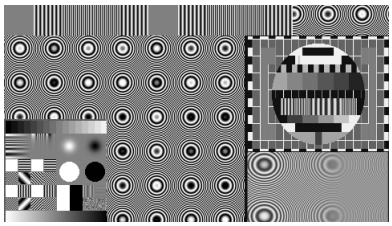
(d)



(e)

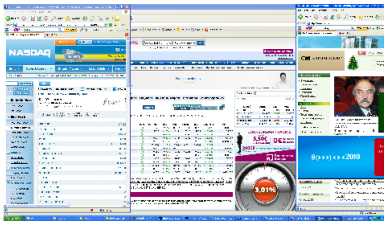


(f)



(g)

(h)



(i)

Figure 50: Set of test images: (a) balcony, (b) rain, (c) car, (d) freeway, (e) plants, (f) fish, (g) testpic, (h) barcelona, (i) boursorma.

A.3 Image Results

In this appendix all the numeric results of applying the CODEC developed in this thesis to the complete set of images are shown. Results of three different QP are calculated. Tables notation is the following: $Width$ and $Height$ are the different sides of the image in pixels. $Bit\ Depth$ is the amount of bits per sample and component each pixels has. $Size$ is the bit size of the original image component data ($Width \times Height \times Bit\ Depth$), $C.Size$ is the correspondent compressed size of the image component after applying compression, $Ratio$ is the compression ratio, $PSNR$ is the Peak Signal to Noise Ratio of each component (Y, U, V), Max is the maximum pixel difference between the original one and the restored pixel, Ave is the average pixel difference considering all of them in each plane, and $Amount$ is the amount of pixels which restored value is not exactly the original one. The last row in the tables named $Total$ contains the total mean values of the sizes and ratios in the complete frame (considering all the three components).

A.3.1 QP=18

car									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	961	2.63	55.15	8	1.39	1614199
Height	1080	U		805	3.15	55.22	8	1.38	1611290
Bit Depth	10	V		792	3.20	55.22	8	1.38	1610709
		Total	7594	2559	2.97				

Table 24: car, $QP = 18$

barcelona									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	914	2.77	55.29	9	1.36	1597410
Height	1080	U		547	4.63	56.24	8	1.21	1542658
Bit Depth	10	V		581	4.36	56.06	8	1.24	1555223
		Total	7594	2043	3.72				

Table 25: barcelona, $QP = 18$

boursorma									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	515	4.92	59.79	9	0.58	830153
Height	1080	U		254	9.97	60.82	9	0.48	693679
Bit Depth	10	V		256	9.89	60.82	9	0.48	696618
		Total	7594	1026	7.40				

Table 26: boursorma, $QP = 18$

freeway									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	866	2.92	55.67	8	1.29	1556625
Height	1080	U		623	4.06	55.99	8	1.24	1544734
Bit Depth	10	V		687	3.69	55.73	9	1.29	1562748
		Total	7594	2176	3.49				

Table 27: freeway, $QP = 18$

balcony									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	800	Y	704	270	2.61	68.74	30	1.00	280411
Height	600	U		170	4.14	68.87	9	0.99	279266
Bit Depth	12	V		138	5.10	69.04	8	0.96	275191
		Total	2110	579	3.64				

Table 28: balcony, $QP = 18$

fish									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	3038	876	3.47	64.30	30	1.25	1485499
Height	1080	U		648	4.69	69.55	9	0.96	1326450
Bit Depth	12	V		654	4.65	69.74	9	0.90	1237088
		Total	9113	2179	4.18				

Table 29: fish, $QP = 18$

testpic									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	1109	2.28	56.47	8	1.08	1340130
Height	1080	U		79	32.05	109.94	1	0.00	22
Bit Depth	10	V		79	32.05	109.94	1	0.00	22
		Total	2532	1268	5.99				

Table 30: testpic, $QP = 18$

plants									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	3038	1816	1.67	53.85	30	3.50	1620814
Height	1080	U		1346	2.26	68.36	8	1.13	1392170
Bit Depth	12	V		1132	2.68	68.32	9	1.13	1395156
		Total	9113	4295	2.12				

Table 31: plants, $QP = 18$

rain									
QP	18	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	800	Y	704	350	2.01	65.36	30	1.37	361916
Height	600	U		126	5.59	70.24	8	0.76	227636
Bit Depth	12	V		143	4.92	70.11	10	0.77	229766
		Total	2110	620	3.40				

Table 32: rain, $QP = 18$

A.3.2 QP=21

car									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	832	3.04	52.47	11	1.92	1736079
Height	1080	U		665	3.81	52.58	11	1.89	1730976
Bit Depth	10	V		653	3.88	52.56	12	1.90	1732684
		Total	7594	2150	3.53				

Table 33: car, $QP = 21$

barcelona									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	782	3.24	52.69	12	1.86	1715143
Height	1080	U		450	5.63	54.05	11	1.57	1653406
Bit Depth	10	V		478	5.30	53.81	11	1.62	1669525
		Total	7594	711	4.44				

Table 34: barcelona, $QP = 21$

boursorma									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	481	5.26	56.99	12	0.88	1087358
Height	1080	U		227	11.15	58.26	11	0.73	973086
Bit Depth	10	V		227	11.15	58.20	12	0.75	1003383
		Total	7594	936	8.11				

Table 35: boursorma, $QP = 21$

freeway									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	751	5.85	53.30	12	1.71	1666079
Height	1080	U		515	4.92	53.74	11	1.62	1649433
Bit Depth	10	V		574	4.41	53.39	12	1.70	1676212
		Total	7594	1841	4.12				

Table 36: freeway, $QP = 21$

balcony									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	800	Y	704	250	2.82	66.02	30	1.41	312103
Height	600	U		149	4.72	66.22	11	1.36	304972
Bit Depth	12	V		118	5.97	66.47	13	1.31	300183
		Total	2110	518	4.07				

Table 37: balcony, $QP = 21$

fish									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	3038	799	3.80	63.29	30	1.60	1594401
Height	1080	U		579	5.25	67.06	13	1.30	1484320
Bit Depth	12	V		588	5.17	67.29	13	1.20	1359934
		Total	9113	1967	4.63				

Table 38: fish, $QP = 21$

testpic									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	1020	2.48	53.97	11	1.48	1504511
Height	1080	U		79	32.05	60.20	1	1.00	2073600
Bit Depth	10	V		79	32.05	60.20	1	1.00	2073600
		Total	7594	1178	6.45				

Table 39: testpic, $QP = 21$

plants									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	3038	1705	1.78	53.71	30	3.95	1733912
Height	1080	U		1251	2.43	65.72	12	1.54	1516815
Bit Depth	12	V		1035	2.94	65.72	13	1.53	1511849
		Total	9113	3992	2.28				

Table 40: plants, $QP = 21$

rain									
QP	21	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	800	Y	704	324	2.17	63.60	30	1.85	390465
Height	600	U		112	6.29	67.80	13	1.01	247361
Bit Depth	12	V		128	5.50	67.61	13	1.04	254858
		Total	2110	564	3.74				

Table 41: rain, $QP = 21$

A.3.3 QP=24

car									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	693	3.65	49.61	16	2.69	1831208
Height	1080	U		521	4.86	49.89	17	2.60	1821883
Bit Depth	10	V		506	5.00	49.82	16	2.62	1824234
		Total	7594	1722	4.41				

Table 42: car, $QP = 24$

barcelona									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	648	3.90	50.02	17	2.53	1801832
Height	1080	U		366	6.92	51.79	15	2.05	1743146
Bit Depth	10	V		385	6.57	51.50	15	2.13	1758626
		Total	7594	1399	5.43				

Table 43: barcelona, $QP = 24$

boursorma									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	444	5.70	54.49	16	1.18	1191391
Height	1080	U		198	12.79	55.87	17	0.98	1124593
Bit Depth	10	V		201	12.60	55.93	18	0.95	1067212
		Total	7594	845	8.99				

Table 44: boursorma, $QP = 24$

freeway									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	639	3.96	50.83	16	2.26	1746887
Height	1080	U		414	6.12	51.39	16	2.11	1730155
Bit Depth	10	V		465	5.45	50.99	17	2.23	1757880
		Total	7594	1519	5.00				

Table 45: freeway, $QP = 24$

balcony									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	800	Y	704	229	3.07	63.15	30	1.98	337001
Height	600	U		128	5.50	63.49	16	1.87	326031
Bit Depth	12	V		98	7.18	63.84	16	1.77	317454
		Total	2110	456	4.63				

Table 46: balcony, $QP = 24$

fish									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	3038	724	4.20	61.79	30	2.11	1697724
Height	1080	U		513	5.92	64.51	17	1.71	1573976
Bit Depth	12	V		524	5.80	64.66	18	1.61	1453937
		Total	9113	1762	4.63				

Table 47: fish, $QP = 24$

testpic									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	2532	931	2.72	51.19	16	2.06	1504511
Height	1080	U		79	30.05	inf	1	-	-
Bit Depth	10	V		79	30.05	inf	1	-	-
		Total	7594	1089	6.97				

Table 48: testpic, $QP = 24$

plants									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	1920	Y	3038	1590	1.91	53.44	30	4.59	1820517
Height	1080	U		1153	2.63	62.92	18	2.11	1604841
Bit Depth	12	V		935	3.25	62.93	17	2.11	1598087
		Total	9113	3679	2.48				

Table 49: plants, $QP = 24$

rain									
QP	24	Plane	Size(KB)	C.Size (KB)	Ratio	PSNR(dB)	Max Δ	Ave Δ	Amount
Width	800	Y	704	296	2.38	63.60	30	2.54	413080
Height	600	U		97	7.26	67.80	19	1.35	262223
Bit Depth	12	V		114	6.18	67.61	17	1.39	267210
		Total	2110	508	4.15				

Table 50: rain, $QP = 24$

References

[1] C. Shannon: Communication theoryexposition of fundamentals. IEEE Journal, 1:4447, February 1953.

[2] David A. Huffman: A method for the construction of minimum-redundancy codes. Proceedings of the I.R.E., 1:10981101, September 1952.

[3] Robert F. Rice and James R. Plaunt: Adaptive variable-length coding for efficient compression of spacecraft television data. IEEE Transactions on Communication Technology, 19(6):889897, December 1971.

[4] Paul G. Howard and Jeffrey Scott Vitter: Arithmetic coding for data compression. Proceedings of the IEEE, 82(6):857865, June 1994.

[5] Raymond Veldhuis and Marcel Breeuwer: An Introduction to Source Coding. Prentice Hall, 1993.

[6] Gilbert Strang: The Discrete Cosine Transform to appear in Society for Industrial and Applied Mathematics (SIAM) Review, 1999.

[7] Hari Kalva, Jae-Beom Lee: The VC-1 and H.264 Video Compression Standards for Broadband Video Services, Springer, 2008.

[8] David Salomon: Data Compression: The Complete Reference, Appendix H, 2nd Edition, Springer ,2000.

[9] H. S. Malvar and G. J. Sullivan: Transform, Scaling & Color Space Impact of Professional Extensions, ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6 Document JVT-H031, Geneva, May 2003.

[10] H. S. Malvar and G. J. Sullivan: YCoCg-R: A Color Space with RGB Reversibility and Low Dynamic Range,ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6 Document JVT-I014, Trondheim, July 2003.

[11] Kees Jaspers: Dimensions of 3D colour space, New Algorithms and their 3D vector based analysis, november 2005.

[12] Yong-Hwan Kim, Byeongho Choi, and Joonki Paik: High-Fidelity RGB Video Coding Using Adaptive Inter-Plane Weighted Prediction, IEEE Transactions on circuits and systems for Video Technology, Vol. 19, No. 7, July 2009.

[13] G. Sullivan: Approximate theoretical analysis of RGB to YCbCr to RGB conversion error, JVT, San Diego, CA, Doc. JVT-I017, Sep. 2003.

[14] Y. H. Kim, J. H. Park, J. W. Kim, B. Choi, and J. Paik: Efficient RGB video coding using adaptive inter-plane residual prediction, Digest of Technical Papers International Conference on Consumer Electronics (ICCE), Las Vegas, NV, Jan. 2008 pp. 12.

[15] Philips file standard for pictorial data tool kit. Available from World Wide Web: http://pfspd. sourceforge.net

[16] O. Eerenberg, Y. Gao, E.Trauschke, P.H.N. de With: Pattern-Based de-correlation for visual-lossless video compression for wireless display applications, Digest of Technical Papers International Conference on Consumer Electronics (ICCE), Las Vegas, NV, Jan. 11 - 13, 2010, pp. 229 - 230.

[17] Iain Richardson: The H.264 Advanced Video Compression Standard, John Wiley & Sons, 2010. Reproduced with permission of Iain Richardson

[18] Joint Video Team of ITU-T and ISO/IEC: Draft Text of H.264/AVC Fidelity Range Extensions Amendment, Doc. JVT-L047, Sept. 2004.

