Politechnika Poznańska
Wydział Elektryczny
Instytut Elektroniki i Telekomunikacji
Ul Piotrowo 3A, 60-965 Poznań

Krzysztof Rakowski

Accumulation of Errors Caused by Image Compression and Color Transformation

Doctoral Dissertation

Advisor: Prof Marek Domański

Poznań, 2004

Politechnika Poznańska
Wydział Elektryczny
Instytut Elektroniki i Telekomunikacji
Ul Piotrowo 3A, 60-965 Poznań

Krzysztof Rakowski

Kumulacja błędów powstających w trakcie kompresji obrazów i transformacji barw

Rozprawa Doktorska
Przedłożona Radzie Wydziału Elektrycznego
Politechniki Poznańskiej

Promotor: Prof. dr hab. inż. Marek Domański

Poznań, 2004

ACKNOWLEDGEMENTS

This dissertation is a result of the fruitful cooperation with my advisor Prof. Marek Domański, who was the spiritus movens for my activity in image compression field. He has introduced me into this subject, showed the basic ideas and theorems and encouraged me to continue my work.

I must not forget about my father Marian who was both the first and the best mathematics teacher to me, my mother Jadwiga who inspired me to learn English and my brother Leszek who introduced me into programming to say nothing about their love and care.

I would like to thank to all my colleagues, who shared their knowledge with me. Maciej Bartkowiak and Adam Łuczak are those, who helped me most. Discussions with them gave me deeper insight into digital data processing, and offered me a chance to understand better how the numbers embodied into processors create the virtual world and improve the virtually real one.

I would like to express my deep gratitude to all those who stand by me and help to overcome difficulties and are not mentioned above.

CONTENTS

Symbols index……....5

I. Introduction……..……...……...……9

1.1 Scope……..9

1.2 Goal, thesis and methodology……...10

1.3 Dissertation overview……...……11

II. Image and video compression……..……13

2.1 Color transformation in compression……13

2.2 Lossless image compression ……...……...……13

2.3 JPEG – transform coding of images……..15

2.4 JPEG2000 – wavelet coding of images……..16

2.5 Hybrid video coding ……17

III. Literature review……....19

3.1 Transformation reversibility……...……...…….19

3.1.1 S-transform…….... 19

3.1.2 S-P transform ……....20

3.1.3 TS-transform……21

3.1.4 Lifting scheme……...22

3.1.5 Transformation matrix factorization – 2D case……..…….....23

3.1.6 Factorization of a transformation matrix – general approach ……....24

3.1.7 Reversible transform coding – alternative approach……..27

3.2 Error accumulation in JPEG, M-JPEG, MPEG……..……..29

IV. Linear transforms – theoretical analysis……...……...33

4.1 Operation of rounding – discussion…….…....33

4.2 Linear transformation – definitions necessary for multiple encoding/decoding cycles analysis……...35

4.3 Linear transformation – necessary condition for its reversibility …….……..38

4.4 Linear transformation – sufficient condition for its reversibility and maximum rounding error in a single cycle for the irreversible one…….……43

4.5 Conditions for error accumulation termination…….....47

4.6 Error accumulation in multiple encoding/decoding cycles……....….50

4.6.1 Recursive formulae for an error evolution ……50

4.6.2 Range clipping ……..53

4.6.3 Consecutive cycles analysis – approach 1……...…….....58

4.6.4 Consecutive cycles analysis – approach 2……...…….....69

4.6.5 Consecutive cycles analysis – approach 3…….....77

V. Color transformation……...81

5.1 Properties of 8-bit YC_BC_R color transformation……...……..81

5.2 Error accumulation for YC_BC_R color transformation in the absence of a range clipping...……...82

5.3 Error accumulation for YC_BC_R color transformation in the presence of a range clipping...……...84

5.3.1 Side faces of RGB range cube……..85

5.3.1.1 R_{max} and R_{min} face of RGB range cube…..…….....89

5.3.1.2 G_{max} and G_{min} face of RGB range cube…….....98

5.3.1.3 B_{max} and B_{min} face of RGB range cube…….....106

5.3.2 Edges of RGB range cube ……..……...……...115

5.4 Conclusions……..……..……..…….....121

VI. Lossy JPEG – error accumulation with and without color transformation..….....123

VII. Near-lossless techniques based on predictive compression...…….…..…....129

7.1 Lossy compression based on linear prediction……..…….…….....129

7.2 Near lossless JPEG-LS error accumulation……..…….…….….129

7.3 New method of near-lossless still image compression with no error accumulation…….……..132

VIII. Video coding...…….....137

IX. Conclusions……...…….....141

References…...……...…….……..145

Symbols index

$x(n)$	n'th sample
$x_o(n)$	n'th odd sample
$x_e(n)$	n'th even sample
$\mathbf{P}\{x(n)\}$	prediction operator
$\lfloor \cdot \rfloor$	downward truncation (floor function)
$\lceil \cdot \rceil$	upward truncation (ceiling function)
$[\cdot]$	rounding to the nearest integer
$\widetilde{h}(n)$	prediction of n'th sample
$\mathbf{U}\{x(n)\}$	update operator
$\mathbf{y}^T$	T denotes transposition of vector/matrix **y**
$\mathbf{y}^\tau$	τ denotes transposition and conjugation of vector/matrix **y**
$\vec{\mathbf{X}}$	data vector in X frame of reference
$\vec{\mathbf{X}}^n$	n-th cycle data vector in X frame of reference with integer components
$\vec{\hat{\mathbf{X}}}^n$	n-th cycle data vector in X frame of reference with real components
$\vec{\mathbf{Y}}$	data vector in Y frame of reference
$\vec{\mathbf{Y}}^n$	n-th cycle data vector in Y frame of reference with integer components
$\vec{\hat{\mathbf{Y}}}^n$	n-th cycle data vector in Y frame of reference with real components
$\mathbf{T}$	forward transformation matrix
t_{ij}	an element of a **T** matrix
$\mathbf{T}_{abs}$	matrix consisting of the absolute values of the **T** matrix elements
$\mathbf{S}$	backward transformation matrix
s_{ij}	an element of an **S** matrix
$\mathbf{S}_{abs}$	matrix consisting of the absolute values of the **S** matrix elements

$\text{round}(\ )$ rounding operation

$\text{clip}(\ )$ range clipping operation

$\text{cut}(\ )$ cutting operation, cutting of increments is equivalent to clipping of components

$\text{sign}(\ )$ sign function – returns the sign of its argument

$\delta^n\vec{\hat{\mathbf{Y}}} = \vec{\mathbf{Y}}^{n+1} - \vec{\hat{\mathbf{Y}}}^n$ increment of $\vec{\hat{\mathbf{Y}}}^n$ - rounding error

$\delta^n\vec{\hat{\mathbf{X}}} = \vec{\mathbf{X}}^n - \vec{\hat{\mathbf{X}}}^n$ increment of $\vec{\hat{\mathbf{X}}}^n$ - rounding error

$\delta^n\vec{\mathbf{Y}} = \vec{\hat{\mathbf{Y}}}^n - \vec{\mathbf{Y}}^n$ increment of $\vec{\mathbf{Y}}^n$ - rounding error

$\delta^n\vec{\mathbf{X}} = \vec{\hat{\mathbf{X}}}^{n+1} - \vec{\mathbf{X}}^n$ increment of $\vec{\mathbf{X}}^n$ - rounding error

$\delta\ \vec{\mathbf{X}}^{\max}$ the greatest possible $\delta^n\vec{\mathbf{X}}$

$\delta R^{\max}$, $\delta G^{\max}$, $\delta B^{\max}$ upper bounds for *RGB* errors

$\delta Y^{\max}$, $\delta C_B^{\max}$ and $\delta C_R^{\max}$ upper bounds for YC_BC_R errors

C_k tells how far away from the edge of the range cuboid its data point is located

$|\ |$ an absolute value

$\det(\ )$ a determinant

$\det_G(\ )$ a Gram determinant

$\mathrm{L}_\infty(\ )$ an infinity norm

$\delta^n\vec{\mathbf{X}}^{Mig1} = \text{round}\left(\delta^n\vec{\mathbf{X}}\right)$ a primary migration vector, - represents the migration between the X elementary cell centers

$\delta^n\vec{\hat{\mathbf{X}}}^{Mig2} = \mathbf{T}^{-1}\cdot\text{round}\left(\mathbf{T}\cdot\ \delta^n\vec{\hat{\mathbf{X}}}\right)$ a secondary migration vector in X frame of reference

$\delta^n\vec{\mathbf{Y}}^{Mig2} = \text{round}\left(\mathbf{T}\cdot\ \delta^n\vec{\hat{\mathbf{X}}}\right)$ a secondary migration vector in Y frame of reference

DCT Discrete Cosine Transform

DCT Discrete Cosine Transform matrix

qDCT	quantized Discrete Cosine Transform matrix
Q	quantization matrix rewritten to match dimensionality
P	permutation matrix
3DCT	DCT matrix for three components
64T	color transformation matrix for 64 pels
PSNR	Peak Signal to Noise Ratio
K	compression ratio

I. INTRODUCTION

1.1 Scope

As the raw visual data are resource consuming, the image and video compression is a vital problem in numerous applications. Either the available disc space or the telecommunication channel capacity inflicts limitations upon the bit-stream. A reduced size bit-stream, produced during compression, may be precise enough to enable the error free data reconstruction. Such compression is called **lossless**. When the reduced size bit-stream precision is sufficient only for the coarse reconstruction of the data, then such a compression is called **lossy**. Hence the lossless compression paradigm is to decrease the statistical redundancy in the signal. The lossy compression paradigm is to decrease the subjective redundancy, which means that the subjectively insignificant data are to be removed from the signal. Usually a human observer is the final receiver of the visual data, hence the human visual system characteristics (HVS) must be taken into account in such a compression.

There are few measures defined in order to assess the compression efficiency. The basic ones are as follows: number of bits in the resulting bit-stream per one symbol of the original signal (bps), the number of bits in the resulting bit-stream per one pixel of the original image (bpp) and the compression ratio. For lossy compression the distortion measure must be also defined. The most common objective measures of distortion are signal to noise ratio (SNR), peak signal to noise ratio (PSNR), mean square error (MSE) and sum of absolute differences (SAD). Unfortunately, such objective measures of distortion are sometimes not consistent with human impression of visual data quality. In most cases however, they are satisfactory. Hence when comparing the efficiency of two different lossy compression algorithms one must take into account both the distortion and the compression

measure whereas when comparing two different lossless compression algorithms it is enough to take into account the compression measure only.

Sometimes there exists a need for processing data after they have been compressed. Usually it is necessary to decompress the bit-stream in order to make such a processing possible. The consecutive lossy compression/decompression cycles may cause the compression errors accumulation. Hence, even if a single compression/decompression cycle does not cause any visible distortion, a sufficiently long sequence of encoding/decoding cycles may produce visible artifacts. In some applications this is not acceptable. A key example is medical image compression since the vital decisions are made on the bases of such images. Even slight changes in colors may be a wrong clue for the doctor. Another example is the visual data processing which is done in TV studios. The compressed material, stored in the archives, may be again used in production of another material. This can be done several times and this is why the problem of error accumulation must be carefully studied.

1.2 Goal, thesis and methodology

Image and video compression is rapidly developing in recent years. The variety of the techniques is astonishing. It is difficult to refer to all of them. However there is one class of the most commonly used techniques. They are based on linear transformations. Such transformations are immanent for transform coding, but are also used as predictors in predictive coding, let alone color transformations. Hence the techniques based on the linear transformations will be studied in this thesis.

The goal of this dissertation is to explore the error accumulation during multiple visual data encoding/decoding cycles. The color transformations, which are simple cases of a

linear transformation, are considered analytically. More complex problems are examined experimentally. A method, which enables restriction of error accumulation while multiple image compression/decompression, is proposed. The most common compression techniques are considered. Numerical analysis is exemplified by images with 8-bit representation of component sample, which is the most common for images and video of natural scenes.

The thesis of the dissertation is following. There exist analytic conditions which can be used in order to examine error accumulation in compression and/or color transformation. In the dissertation the conditions are formulated and used for some cases. The problems, which are to complex for theoretical analysis, are examined experimentally.

1.3 Dissertation overview

The dissertation consists of three parts. The first one presents the overview of the knowledge and problems which are related to the dissertation subject and the first three chapters constitute it. Recently developed techniques for reversible transform coding are briefly described in Section 3.1. Section 3.2 summarizes the articles on an experimental analysis of error accumulation in MPEG-2 and JPEG coders.

The second part is a theoretical analysis of the linear transformations and consists of Chapter IV and V. The most interesting original achievements are presented in this part. Theoretical analysis of a range clipping is something that was not carefully studied so far.

The final three chapters present the experimental results and the alternative technique for near-lossless coding of still images. The problems approached in this chapter are so complex, that the most interesting conclusions are drawn on the experimental bases.

II. IMAGE AND VIDEO COMPRESSION

2.1 Color transformation in compression

As color images and video sequences are in common use, the color transform is vital. Compressing every original (red, green and blue) color component separately may be utterly inefficient. This is because the red, green and blue components, which are most commonly used in the image acquisition devices, are highly correlated. It is very profitable to remove this correlation before the very compression is done. The linear transformation which removes this correlation is Karhunen-Loeve Transform (KLT) [Bovik], [Gray], [Jajant]. It is an image specific transform and one must include its coefficients into the compressed data. This may be costly and the common practice is to use a relatively efficient sub-optimal color transformation which is a priori known both to the encoder and decoder. The RGB -> YC_BC_R color transformation is probably the one which is most commonly used. Its de-correlation efficiency is very close to the optimal one and this is why it has been applied in many standards, including JPEG [JPEG_1] [JPEG_2] [JPEG_3], JPEG2000 [JPEG2000], H.261 [H261], H.263 [H.263], MPEG-1 [MPEG1], MPEG-2 [MPEG2] and many others.

2.2 Lossless image compression

The majority of techniques for lossless compression of images are based on predictive coding. The JPEG-LL [JPEG_3], JPEG-LS [JPEG_LS] and CALIC [Wu1] [Wu2] [Bovik] are the most widespread standards. The common idea of all these techniques is to predict the incoming symbols using those, which have already been encoded. Then the prediction error is entropy coded. In order to make the compression efficient, the predictor must be appropriate for the encoded signal. The main idea is to create a new alphabet, which has the

PMF (Probability Mass Function) as diverse as possible. If this is achieved then entropy coding will produce a well compressed bit-stream.

The oldest standard is JPEG lossless mode (JPEG-LL). It works either with no prediction or with one of seven defined predictors. Most applications use only one of those seven linear predictors for the whole scan with limitations on image boundary. The resulting prediction errors are fed either to Huffman or to arithmetic coder.

CALIC is a one pass coder. It is considered as the state-of-the-art in lossless image coding and is rather complex. The prediction is done in two stages. The initial prediction uses one of the six linear predictors but unlike the JPEG-LL the CALIC encoder switches between them adapting to the signal. This adaptation is done according to the neighborhood of the currently encoded pixel. Various sets of values in the neighboring pixels are grouped into 576 pixel contexts. For every context the average value of the prediction error is found. The initial prediction is then modified by the above mentioned average. Then the difference between the final prediction and the actual value of the pixel is further coded. Again eight contexts for the prediction error are found. Then those errors are remapped and entropy coded with regard to the prediction error contexts.

The JPEG-LS encoder is more similar to the CALIC coder then the earlier lossless JPEG (JPEG-LL). It is simpler then CALIC but more sophisticated then JPEG-LL. It is also a one-pass coder. Its efficiency is surprisingly good. In this standard also the prediction is done in two stages. In the first stage one of the three predictors is chosen. Two of them are non-linear and the third one is linear. The encoder switches between various predictors adapting to the incoming data and produces the initial prediction. The contexts are also created and the average prediction error is found for each of them. Then it is used to create

the final prediction. The prediction error, after re-mapping, is entropy coded. The adaptive Golomb codes are used.

Both CALIC and JPEG-LS have modes of near-lossless operation. In this mode the maximum difference between the original and the decoded image is controlled. This is done by quantization of the prediction error.

2.3 JPEG - transform coding of images

The most widespread standard for lossy image coding is JPEG [JPEG_1] [JPEG_2] [JPEG_3] [Pennebaker] [Bovik]. Its main idea is the block-wise discrete cosine transform (DCT) which is computed in 8×8 pixel blocks. It is done on de-correlated image components with chrominance optionally decimated. Such decimation is justified by the characteristics of the human visual system. The resulting DCT coefficients are then quantized with appropriately scaled quantization tables. Usually the quantization step sizes are greater for higher frequencies in both horizontal and vertical direction. This is the main rate-distortion control mechanism. The coarser quantization is applied the greater compression is achieved. For high compression ratios however, the annoying coding artifacts become visible. These are for example color distortions, block and ringing effects. The 8×8 block of quantized coefficients is then ordered into a one-dimensional sequence, with the lowest frequencies at the beginning and the highest at the end. The coarser quantization is applied the more zero valued coefficients are found in this sequence. Hence the runs of zeroes together with the first non-zero coefficient are coded with run-length coding. The resulting numbers are entropy coded producing the final bitstream.

2.4 JPEG2000 - wavelet coding of images

The new standard JPEG 2000 [JPEG2000] [Marcelin] utilizes the wavelet technique for image compression. There are two wavelet transforms defined in the standard. The first one is realized with a 5/3 filter bank [Tabatabai], and the second one with a 9/7 filter bank [Daubechies]. Both these banks are constructed according to the lifting principle. This first one is lossless and combined with lossless color transform creates the lossless path for image compression. The fine granularity scalability is supported in this standard, which is achieved with the EBCOT algorithm [Taubman1] [Taubman2]. The data are optionally partitioned into rectangle tiles of equal sizes. Every such a tile is independently coded. The greatest possible tile size is the size of the whole image.

If the image consists of three components only, which is typical for the pictures of natural scenes, then color transform will be used. In the lossy path of coding the RGB -> YC_BC_R color transformation is used, and in the lossless path the special lossless color transform is used. Such transformation is used in order to remove the correlation between the color components. Subsequently either lossless or lossy wavelet transform is done, depending on the mode of compression. The resulting transform samples can be quantized in the lossy path. The coefficients are grouped into rectangle blocks of equal sizes, called "code-blocks". Every such a code block is independently entropy coded, which is done bit-plane by bit-plane. The bits of every bit-plane are collected in three passes and fed to the arithmetic encoder, which produces an elementary bit-stream. The EBCOT paradigm is to truncate every elementary bit-stream into few quality layers. These truncation points are chosen with the Lagrange optimization method. Hence the elementary bit-stream fragments, which belong to the lowest quality layer, are concatenated to initiate the total bit-stream. Then the elementary bit-stream fragments, which belong to the next quality layer, are

concatenated and appended to the total bit-stream. This operation is repeated until all the elementary bit-stream fragments are included in the total bit-stream. Hence the highly scalable compressed bit-stream originates. If the lossless path is used, then the scalability ranges from the lowest quality level image up to ideally reconstructed image. This is a novel feature of the JPEG2000 standard, which incorporates the lossy to lossless compression in one scheme.

2.5 Hybrid video coding

The majority of video coders are based on the transform coding and the discrete cosine transform in 8×8 blocks is the one which is most commonly used. Such standards like [H261], [H263], [MPEG] and [MPEG2] are based on it. They all follow the same paradigm yet they differ in details as they were designed to match different demands. Standard [H261] is suitable for video-conference systems which are built over ISDN networks. Standard [H263] is similar to the latter one, yet it is designed for video telephony, hence it must cope with smaller telecommunication channel capacity. Standard [MPEG1] was intended to be used for films storage on compact discs. Standard [MPEG2] became widespread in all the areas dealing with digital television. All these standards group the four neighboring blocks of pixels into one macroblock, which is used for motion compensation. Macroblock can be an I, P or B type. In the standard [H261], which is the oldest one, only I and P type macroblocks are allowed. I type means that the very image samples are encoded. P and B types imply that the prediction is done and the resulting prediction errors are used in DCT calculations. P type means that the macroblock from previous frame is used for prediction and B type means that the two macroblocks, one from the preceding and one from the subsequent frame, may be used for prediction. The macroblock used for prediction does

not need to have the same position in a frame as the predicted macroblock. If motion compensation is used, then the motion vectors will have to be found in order to show which macroblock in the reference frame is the best match. Motion compensation is optional in [H261] standard and obligatory in others. The DCT coefficients are quantized. The first two standards use the same quantizer for all the coefficients and the last two standards take into account the human visual system when quantizing the coefficients. Quantization step size can be modified while coding the video-sequence in order to make the rate-distortion control possible. In order to prevent error accumulation when encoding the video sequences all the standards introduce some precautions. The first two demand the I type macroblock to appear in the stream with some minimum frequency. The second two introduce the structure called Group of Pictures. All the video sequence must be divided into such GOP's. Each GOP must consist of one I frame and some other type frames. I frame contains only I type macroblocks. Such an organization of data has another advantage – enables random access. Quantized coefficients are ordered into a sequence. As many zero valued coefficients occur in it the run length coding is applied. The length of zero run together with the value of the first non-zero coefficient is mapped into a number. Those numbers are entropy coded.

III. LITERATURE REVIEW

3.1 Transformation reversibility

In order to approach the problem of error accumulation while multiple image encoding and decoding one must study the reversibility of the transformations applied. A transform can be considered **lossless** or **reversible** only if the superposition of the forward transform, followed by rounding and the inverse transform, followed by rounding is the identity transformation. Reversibility of the linear transforms was carefully studied recently. The research on it started when first attempts were made to invent reversible wavelet transforms.

3.1.1 S-transform

S-transform was the first step in this evolution. It was published by H. Blume and A. Fand [Blume1], [Heer1]. S-transform is presented below after A. Said and W. A. Pearlman [Said1].

$$l(n) = \left\lfloor \frac{c(2n) + c(2n+1)}{2} \right\rfloor, \quad n = 0,\ldots,\frac{N}{2} - 1$$
$$h(n) = c(2n) - c(2n+1), \quad n = 0,\ldots,\frac{N}{2} - 1 \qquad \textbf{(3.01)}$$

where $\lfloor \cdot \rfloor$ denotes downward truncation, which is used to remove redundancy. As both sum and difference of two integers is either odd or even, one can decide if not truncated $l(n)$ is integer or not by checking the $h(n)$ parity. The inverse transform is as follows:

$$c(2n) = l(n) + \left\lfloor \frac{h(n)+1}{2} \right\rfloor,$$
$$c(2n+1) = c(2n) - h(n). \qquad \textbf{(3.02)}$$

To prove the first one of the two above formulae one must consider all the cases of $h(n)$ and $l(n)$ parity separately. Proof for the second one is obvious. The S-transform is similar

to the Haar multi-resolution representation and the only difference is the truncation operation. Versions given in a literature may differ from the one given above by some minor details. S-transform as a transformation of frame of reference in a two dimensional samples space (odd and even indexed sample constitute a vector) followed by rounding is presented by Steven Dewitte and Jan Cornelis in [Dewitte1].

3.1.2 S-P transform

The S transform was further developed by A. Said and W. A. Pearlman [Said1]. They invented the **S-P transform**. They aimed at reduction of a residual correlation between the highpass components, which is due to aliasing from the low-frequency components of the original image. Their idea is to use prediction technique (S-transform and Prediction). However, instead of using prediction in the final S-transformed pyramid, during each 1-D transformation they use some low and high-pass filter output samples to predict high pass filter output sample. The low pass samples and prediction errors for high pass samples (the differences between the predictions of high pass samples and their actual values) are entropy coded. This can be written as follows.

$$h_d(n) = h(n) - \left\lfloor \tilde{h}(n) + \frac{1}{2} \right\rfloor, \quad n = 0,1,...,\frac{N}{2} - 1, \tag{3.03}$$

where $h(n)$ is the actual value of the high-pass filter output sample and $\tilde{h}(n)$ is its prediction. This prediction is calculated as follows:

$$\tilde{h}(n) = \sum_{i=-L_0}^{L_1} \alpha_i \cdot \Delta l(n+i) - \sum_{j=1}^{H} \beta_j \cdot h(n+j), \tag{3.04}$$

where $\Delta l(n) = l(n-1) - l(n)$

During the inverse 1-D transformation, the prediction can be added following a reverse order:

$$h(n) = h_d(n) + \left\lfloor \tilde{h}(n) + \frac{1}{2} \right\rfloor, \quad n = \frac{N}{2} - 1, \frac{N}{2} - 2, \ldots, 0, \tag{3.05}$$

so that the values of $h(n)$ required to calculate the prediction for the current n have already been recovered. The main point is to use suitable prediction filter coefficients.

3.1.3 TS-transform

Another extension of S-transform is **TS-transform** (or two-six transform named after the number of taps in the low and high pass respectively). It is defined by the expression of the two outputs:

$$\begin{aligned} s(1) &= \left\lfloor \frac{x(1) + x(2)}{2} \right\rfloor, \\ d(1) &= \left\lfloor \frac{-\left\lfloor \frac{x(-1) + x(0)}{2} \right\rfloor + 4 \cdot x(1) - 4 \cdot x(2) + \left\lfloor \frac{x(3) + x(4)}{2} \right\rfloor}{4} \right\rfloor. \end{aligned} \tag{3.06}$$

This filter pair was derived by Said and Pearlman [Said2] as one of the parameterizations of their base equations. These filters can be enhanced by adding 2 to the numerator $d(n)$, which was done by A. Zandi, M. Boliek, E. L. Schwartz and A. Keith. [Zandi1] and [Zandi2]. The expression for $d(n)$ can be simplified and written with the use of $s(n)$ These result in:

$$\begin{aligned} s(1) &= \left\lfloor \frac{x(1) + x(2)}{2} \right\rfloor, \\ d(1) &= x(1) - x(2) + \left\lfloor \frac{-s(-1) + s(3) + 2}{4} \right\rfloor. \end{aligned} \tag{3.07}$$

When comparing the two above formulae with the S-transform one can see that the main difference between S-transform and TS-transform is an additional truncated component in the second equation.

The TS-transform is reversible and the inverse is:

$$x(1) = s(1) + \left\lfloor \frac{d(1) - \left\lfloor \frac{-s(-1) + s(3) + 2}{4} \right\rfloor + 1}{2} \right\rfloor,$$

$$x(2) = s(1) - \left\lfloor \frac{d(1) - \left\lfloor \frac{-s(-1) + s(3) + 2}{4} \right\rfloor}{2} \right\rfloor. \qquad \textbf{(3.08)}$$

3.1.4 Lifting scheme

The **lifting scheme** introduced by Sweldens may consists of the following stages (after Roger L Claypoole [Claypoole1]):

- **split**: divide the original data into two disjoint subsets. The original data $x(n)$ may be divided into odd indexed points $x_o(n)=x(2n+1)$ and even indexed points $x_e(n)=x(2n)$;
- **predict**: generate wavelet coefficients $d(n)$ as the error in predicting $x_o(n)$ from $x_e(n)$ using prediction operator **P**: $d(n)=x_o(n) - \mathbf{P}\{x_e(n)\}$
- **update**: combine $x_e(n)$ and $d(n)$ to obtain coefficients $c(n)$ that represent a coarse approximation to the original signal $x(n)$. This is accomplished by applying an update operator **U** to the wavelet coefficients and adding to $x_e(n)$: $c(n)=x_e(n)$-$\mathbf{U}\{d(n)\}$,

The above mentioned lifting steps are presented in the following figure.

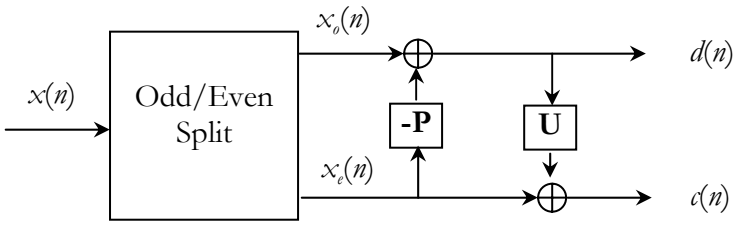

Fig 3.1. Lifting steps forming lifting stage – block diagram.

One can recognize a ladder filter structure in the above figure. These three lifting steps form a *lifting stage*. Iteration of the lifting stage on the output $c(n)$ creates the complete set of DWT scaling and wavelet coefficients $c^j(n)$ and $d^j(n)$. The lifting steps are easily inverted, even if **P** and **U** are nonlinear or space-varying. The inverted lifting steps are as follows:

$$x_e(n) = c(n) - \mathbf{U}\{d(n)\}, \quad x_o(n) = d(n) + \mathbf{P}\{x_e(n)\}. \tag{3.09}$$

Linear operators **P** and **U** are obtained after factorization of the polyphase matrix [Daubechies1], [Gouze1]. Rounding operation must be appropriately done after each step of lifting procedure to achieve reversibility. The conditions which have to be satisfied for the factorization of a transform into lifting steps are to be presented later. Lifting scheme in terms of wavelet theory is a way of generating a new set of biorthogonal filters from a known biorthogonal set. A drawback of the lifting scheme is that it does not guarantee that the wavelet associated with the iterated filter bank exists in $L_2(R)$. In practice this raises the possibility that the filter bank may produce numerical artifacts. This problem is discussed in [McDarby1], [Sweldens1], [Calderbank1].

3.1.5 Transformation matrix factorization - 2D case

According to Bruekers et al. [Bruekers1] a transform with determinant equal one can be represented by a Ladder Network based on PLUV decomposition:

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ (d-1)/b & 1 \end{bmatrix} \begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ (a-1)/b & 1 \end{bmatrix}. \tag{3.10}$$

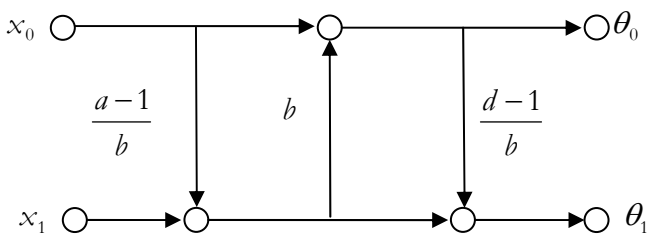

*Fig 3.2. Signal flow graph - Ladder Network realizing matrix **A**.*

The above mentioned decomposition makes the transform reversible. Even if the coefficients and the intermediate results are rounded, the input can be reconstructed losslessly.

3.1.6 Factorization of a transform matrix – general approach

The general approach to the transform reversibility is presented by Pengwei Hao and Qingyun Shi in [Hao]. Few interesting theorems are proved in it. The problem of necessary conditions for factorizing a transform into reversible steps is approached in it. It is inevitable to introduce some definitions first.

Elementary matrix is a matrix which can be expressed as follows:

$$\mathbf{E} = \mathbf{I} \pm \alpha \mathbf{x}\mathbf{y}^{\tau} \qquad (3.11)$$

were **x** and **y** are column vectors, and τ denotes transposition and conjugation.

Unit factor is a one of the four possible numbers: {-i, i, -1, 1}.

ERM – Elementary Reversible Matrix, which is a matrix that corresponds to an elementary reversible structure which enables perfectly invertible integer implementation due to properly arranged computational ordering.

SERM – Single row Elementary Reversible Matrix, which is a matrix with unit factors on its main diagonal, and only one row of nonzero elements.

Unit SERM – Single row Elementary Reversible Matrix, which is an elementary matrix with ones on its main diagonal, and only one row of nonzero elements. Such matrices can be expressed as follows:

$$\mathbf{S}_m = \mathbf{I} + \mathbf{e}_m \mathbf{s}_m^T, \qquad (3.12)$$

where $\mathbf{e}_m$ is the *m*-th standard basis vector formed as the *m*'th column of the identity matrix, and $\mathbf{s}_m$ is a column vector whose *m*-th element is 0.

TERM – Triangular Elementary Reversible Matrix, which is a triangular matrix (strictly speaking it is not an elementary matrix) with unit factors on its main diagonal.

Unit TERM – Triangular Elementary Reversible Matrix, which is a triangular matrix (strictly speaking it is not an elementary matrix) with ones on its main diagonal.

Lifting matrix – is a matrix whose diagonal elements are ones and only one non-diagonal element is nonzero [Zeng].

For 2×2 matrices unit TERM, unit SERM and lifting matrix are the same.

It is clear that if a transformation can be factorized into the matrices being one of the above mentioned types, it will be a perfectly invertible transform, provided the computations are properly arranged. This situation is presented on the following figure.

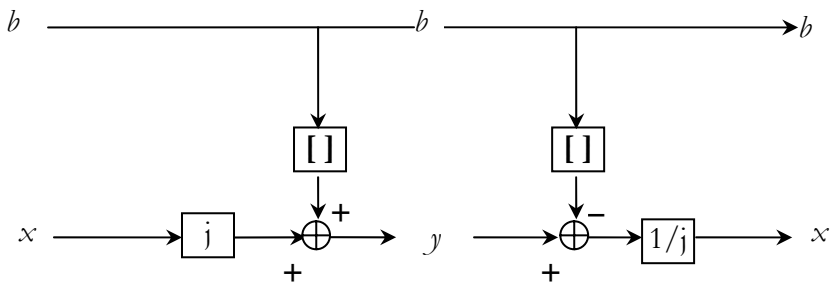

Fig 3.3 Forward and reverse transform of a number implemented by integer mapping..

Symbol [.] denotes either rounding to the nearest integer or truncation and symbol j denotes an integer factor. Operations of forwards and backwards transforms with **TERM** and **SERM** matrices corresponding to the above figure are presented below.

Forward upper **TERM** matrix transform:

$$\begin{cases} y(m) = j_m \cdot x(m) + \left[\sum_{n=m+1}^{N} a_{mn} \cdot x(n) \right] = j_m \cdot x(m) + [b_m], \\ y(N) = j_N \cdot x(N). \end{cases} \quad \textbf{(3.13)}$$

Backward upper **TERM** matrix transform:

$$\begin{cases} x(N) = y(N)/j_N\ , \\ x(m) = (1/j_m)\cdot\left(y(m) - \left[\sum_{n=m+1}^{N} a_{mn}\cdot x(n) \right] \right) = (1/j_m)\cdot\left(y(m) - [b_m] \right). \end{cases} \tag{3.14}$$

Forward lower **TERM** matrix transform:

$$\begin{cases} y(m) = j_m\cdot x(m) + \left[\sum_{n=1}^{m-1} a_{mn}\cdot x(n) \right] = j_m\cdot x(m) + [b_m], \\ y(1) = j_1\cdot x(1). \end{cases} \tag{3.15}$$

Backward lower **TERM** matrix transform:

$$\begin{cases} x(1) = y(1)/j_1\ , \\ x(m) = (1/j_m)\cdot\left(y(m) - \left[\sum_{n=1}^{m-1} a_{mn}\cdot x(n) \right] \right) = (1/j_m)\cdot\left(y(m) - [b_m] \right). \end{cases} \tag{3.16}$$

Forward **SERM** matrix transform:

$$\begin{cases} y(m) = j_m\cdot x(m) + \left[\sum_{n=1}^{m-1} a_{mn}\cdot x(n) + \sum_{n=m+1}^{N} a_{mn}\cdot x(n) \right] = j_m\cdot x(m) + [b_m], \\ y(k) = j_k\cdot x(k) \quad for \quad k \neq m. \end{cases} \tag{3.17}$$

Backward **SERM** matrix transform:

$$\begin{cases} x(k) = y(k)/j_k \quad for \quad k \neq m, \\ x(m) = (1/j_m)\cdot\left(y(m) - \left[\sum_{n=1}^{m-1} a_{mn}\cdot x(n) + \sum_{n=m+1}^{N} a_{mn}\cdot x(n) \right] \right) = (1/j_m)\cdot\left(y(m) - [b_m] \right). \end{cases} \tag{3.18}$$

Pengwei Hao and Qingyun Shi in their article proved appropriate theorems and gave following corollaries for N by N matrix ***A***, where ***P*** is a permutation matrix:

Corollary 1: Matrix ***A*** has a **TERM** factorization of ***A=PVV…V*** if and only if det***A*** is an integer factor.

Corollary 2: Matrix ***A*** has a **unit TERM** factorization of ***A=PVV…V*** if and only if

$$\det\boldsymbol{A} = \det\boldsymbol{P} = \pm 1. \tag{3.19}$$

Corollary 3: Matrix ***A*** has a **TERM** factorization of $\boldsymbol{A}=\boldsymbol{PLUS_0}$ if and only if det$\boldsymbol{A}$ is an integer factor, where ***L*** and ***U*** are lower and upper **TERM** respectively and $\boldsymbol{S_0}$ is **SERM**.

Corollary 4: Matrix ***A*** has a **unit TERM** factorization of $\boldsymbol{A}=\boldsymbol{PLUS_0}$ if and only if det$\boldsymbol{A}$ = det$\boldsymbol{P}$ = ±1, where ***L*** and ***U*** are lower and upper **unit TERM**s respectively and $\boldsymbol{S_0}$ is **SERM**.

Corollary 5: Matrix ***A*** has a **SERM** factorization of $\boldsymbol{A}=\boldsymbol{PS_NS_{N-1}}\ldots\boldsymbol{S_1S_0}$ if and only if det$\boldsymbol{A}$ is an integer factor, where $\boldsymbol{S_m}$ (m = 0, 1, 2,…, N) are **SERM**s.

Corollary 6: Matrix ***A*** has a **unit SERM** factorization of $\boldsymbol{A}=\boldsymbol{PS_NS_{N-1}}\ldots\boldsymbol{S_1S_0}$ if and only if det$\boldsymbol{A}$ is an integer factor, where $\boldsymbol{S_m}$ (m = 0, 1, 2,…, N) are **unit SERM**s.

Hence necessary conditions for the linear transform to be reversible were formulated by the above mentioned authors.

3.1.7 Reversible transform coding - alternative approach

Another approach to the discussed problem was presented by Kunitoshi Komatsu and Kaoru Sezaki [komatsu1]. They do not use decomposition of transform matrices and exploit the characteristics of "floor" function instead.

Their reasoning is cited below:

Let us consider the transform:

$$\begin{bmatrix}\theta(0)\\ \theta(1)\end{bmatrix}=\begin{bmatrix}x(0)+\lfloor c_0\cdot x(1)+0.5\rfloor\\ x(1)+\lfloor c_1\cdot x(0)+0.5\rfloor\end{bmatrix}, \qquad \textbf{(3.20)}$$

where $\theta(0)$ and $\theta(1)$ are integer transform coefficient and $x(0)$ and $x(1)$ are integer inputs. If the real numbers c_0 and c_1 satisfy $c_0c_1 \le 0$ this transform becomes reversible. If the floor functions are omitted, the determinant of the transform matrix becomes $1-c_0c_1$. Therefore, redundancy occurs in transform domain, when $c_0c_1 < 0$. This problem is avoided by using the following transform instead:

$$\begin{bmatrix}\theta(0)\\ \theta(1)\end{bmatrix}=\begin{bmatrix}x(0)+\lfloor c_0\cdot x(1)+0.5\rfloor\\ x(1)+\lfloor c_1\cdot\theta(0)+0.5\rfloor\end{bmatrix}. \tag{3.21}$$

It is obvious that this transform is reversible for any c_0 and c_1. If the floor functions are omitted, the coefficients of θ_1 standing by x_0 and x_1 are c_1 and $1+c_0c_1$, respectively, that is, the determinant becomes 1.

In order to get uniform dynamic range the following reversible transform is proposed:

$$\text{Forward: } \begin{bmatrix}\theta(0)\\ \theta(1)\\ \theta(2)\end{bmatrix}=\begin{bmatrix}x(0)+\lfloor c_0\cdot x(1)+0.5\rfloor\\ x(1)+\lfloor c_1\cdot\theta(0)+0.5\rfloor\\ \theta(0)+\lfloor c_2\cdot\theta(1)+0.5\rfloor\end{bmatrix}; \tag{3.22}$$

$$\text{backward: } \begin{bmatrix}\theta(0)\\ x(1)\\ x(0)\end{bmatrix}=\begin{bmatrix}\theta(2)-\lfloor c_2\cdot\theta(1)+0.5\rfloor\\ \theta(1)-\lfloor c_1\cdot\theta(0)+0.5\rfloor\\ \theta(0)-\lfloor c_0\cdot x(1)+0.5\rfloor\end{bmatrix}. \tag{3.23}$$

where the transform coefficients are $\theta(1)$ and $\theta(2)$. If the floor function is omitted, the coefficients of $\theta(1)$ standing by $x(0)$ and $x(1)$ become c_1 and $1+c_0c_1$, respectively. and those of $\theta(1)$ become $1+c_1c_2$ and $c_0+c_2+c_0c_1c_2$ respectively. Therefore, the determinant becomes one.

The above presented transform can be generalized into N-point transform as follows:

forward transform

$$\begin{aligned}\theta(j)&=x(j)+\left\lfloor\sum_{j=0}^{j-1}c_{i,j}\cdot\theta(i)+\sum_{i=j+1}^{N-1}c_{i,j}\cdot x(i)+0.5\right\rfloor,(0\le j\le N-1),\\ \theta(N)&=\theta(0)+\left\lfloor\sum_{j=1}^{N-1}c_{i,N}\cdot\theta(i)+0.5\right\rfloor,\end{aligned} \tag{3.24}$$

backward transform

$$\begin{aligned}\theta(0)&=\theta(N)-\left\lfloor\sum_{i=1}^{N-1}c_{i,N}\cdot\theta(i)+0{,}5\right\rfloor,\\ x(j)&=\theta(j)-\left\lfloor\sum_{i=0}^{j-1}c_{i,j}\cdot x(i)+0.5\right\rfloor,(N-1\ge j\ge 0),\end{aligned} \tag{3.25}$$

.

where $\sum_{i=0}^{-1} = \sum_{i=N}^{N-1} \equiv 0$ and the transform coefficients are $\theta(1)$, $\theta(2)$,..., $\theta(N)$.

The authors exploited this idea to derive the reversible versions of many commonly known transforms. They firstly obtained the generalized transform matrix form Eq. **(3.24)** and then proposed coefficients for a reversible transform by comparing achieved generalized transform matrix coefficients with desired transform matrix coefficients. Their results are published in the following articles: [komatsu1], [komatsu2], [komatsu3], [komatsu4].

3.2 Error accumulation in JPEG, M-JPEG, MPEG2

The problem of error accumulation while multiple image compression/decompression cycles is not commonly studied. Rather few articles on this topic can be found. Those cited here were published in the mid-nineties when 4 : 2 : 2 Studio Profile of MPEG-2 was being developed. The first one is [Cornog1]. Video test material used for experiments consisted of four video sequences: Billboard, Mobile, Flower Garden and Swanboats concatenated into one video sequence. The digital data was stored in 4:2:2 format. The motion JPEG codec (M-JPEG) and the MPEG-2 4:2:2 Main Level codec compressed the entire 4:2:2 signal. The MPEG-2 Main Profile Main Level codec decimated the input to 4:2:0 prior to compressing it, and interpolated after decompressing. The authors claim JPEG performed slightly better than the MPEG-2 4:2:2 profile I-frame encoder. Unfortunately not much is said about the strategy which was used for bitrate control. The author claims that multi-generation JPEG incurs no additional loss after the first generation of compression, provided that the pixels remain identically aligned, and stay in the same video format. This statement will be discussed in the dissertation more thoroughly. In the same article it is said that under the same set of

conditions, MPEG will incur a small amount of loss on each generation, but most of the loss will occur on the first generations and after that the loss will diminish asymptotically; both in JPEG and MPEG, if any changes are made to pixel values or alignment between generations, a larger loss in quality will occur. Another article which approaches the same problem is [Horne1]. The authors show that the 4:2:2 Profile which extends Main Profile/Main Level is a suitable tool for post processing of video material. It behaves properly when multiple compression/decompression cycles are performed. They made experiments for intra studio environment and inter studio environment. For the first environment intensive post-processing is assumed hence I or IBI GOP structure is chosen. In inter studio environment editability is less of an issue, and longer GOP can be used. The experiments were done for two test video –sequences: "cheer leaders" and "mobile & calendar" both were 524 lines, 60 Hz material. Bitrate control was set to 30 Mb/s and 50 Mb/s. To simulate a hybrid analog/digital environment and/or digital video effects, either a spacial or temporal shifts were used. It was shown that at the highest bit rates, 50 Mb/s, PSNR ranges form 41 dB for a complex sequence as mobile & calendar using GOP structure of all I, up to 44 dB for sequence cheerleaders. This high quality is retained in multi-generation coding environments. The algorithm is robust to multi generation coding. The biggest loss in quality occurs when performing simulated digital video effects, such as spatial or temporal shifting. In such cases , using TM-like encoders, the loss in PSNR over eight generations can go up to almost 5 dB at higher bitrates. The authors also conducted the experiments with M-JPEG (motion JPEG) in order to compare the results with 4:2:2 MPEG2. In those experiments no shifting was performed between the generations. A frame based rate control was used. It was slowly adjusting the quantization factor to maintain a constant average number of bits per picture. The authors show that MPEG2 4:2:2 outperforms M-JPEG also in multiple compression/decompression cycles. Unfortunately nothing is said about M-JPEG

chrominance decimation format but one should expect that it is also 4 : 2 : 2. Concluding the authors say that MPEG2 4:2:2 profile can provide a good to excellent quality in both intra and interstudio environment. It is robust in a multi-generation coding environment with the small degradations in quality diminishing rapidly with increasing generations.

IV. LINEAR TRANSFORMATIONS – THEORETICAL ANALYSIS

The theoretical analysis of N – dimensional linear transforms is presented in Chapter 4. Sections 4.1 and 4.2 are the introductory ones. The properties and definitions introduced there are further used. Sections 4.3 and 4.4 are dealing with the single encoding/decoding cycle. **Encoding** is to be understood here as a forward linear transform followed by rounding and **decoding** – as the inverse linear transform followed by rounding. In Sections 4.5 and 4.6 the analysis of multiple encoding/decoding cycles is presented.

4.1 Operation of rounding - discussion

The rounding operation can be defined in many alternative ways. The "ceiling" ($\lceil\ \rceil$) and "floor" ($\lfloor\ \rfloor$) functions [Graham] are very useful for this purpose. Each definition implies different properties. The three exemplary definitions are presented below:

$$\begin{aligned} &1) \quad \mathrm{round}_1(x) = \lfloor x + 0.5 \rfloor, \\ &2) \quad \mathrm{round}_2(x) = \lceil x - 0.5 \rceil, \\ &3) \quad \mathrm{round}_3(x) = \begin{cases} \lfloor x + 0.5 \rfloor & \textit{for } x \geq 0, \\ \lceil x - 0.5 \rceil & \textit{for } x < 0. \end{cases} \end{aligned} \qquad \textbf{(4.01)}$$

The third definition is in common use because of its convenient property:

$$\mathrm{round}(-x) = -\mathrm{round}(x) \qquad \textbf{(4.02)}$$

for all real x. However this property is not critical in the reasoning conducted in this thesis. Unfortunately, it lacks another property:

$$\mathrm{round}(x + int) = int + \mathrm{round}(x) \qquad \textbf{(4.03)}$$

for all real x and all integer int.

In order to prove this, it is enough to see that the condition **(4.03)** is not satisfied for $x = (2 \cdot n - 1) \cdot 0.5$ and $int = -2 \cdot n \cdot 0.5$, where n is a positive integer number. In such a case the left hand side of the expression **(4.03)** is equal to minus one and the right hand side of the

expression **(4.03)** is equal to zero. The small arrows on Fig. 4.6 represent the influence of the rounding operation No 3 **(4.01)** onto the argument of the operator "round" where L – denotes the left hand side of the expression **(4.03)** and R – denotes the right hand side of the expression **(4.03)**.

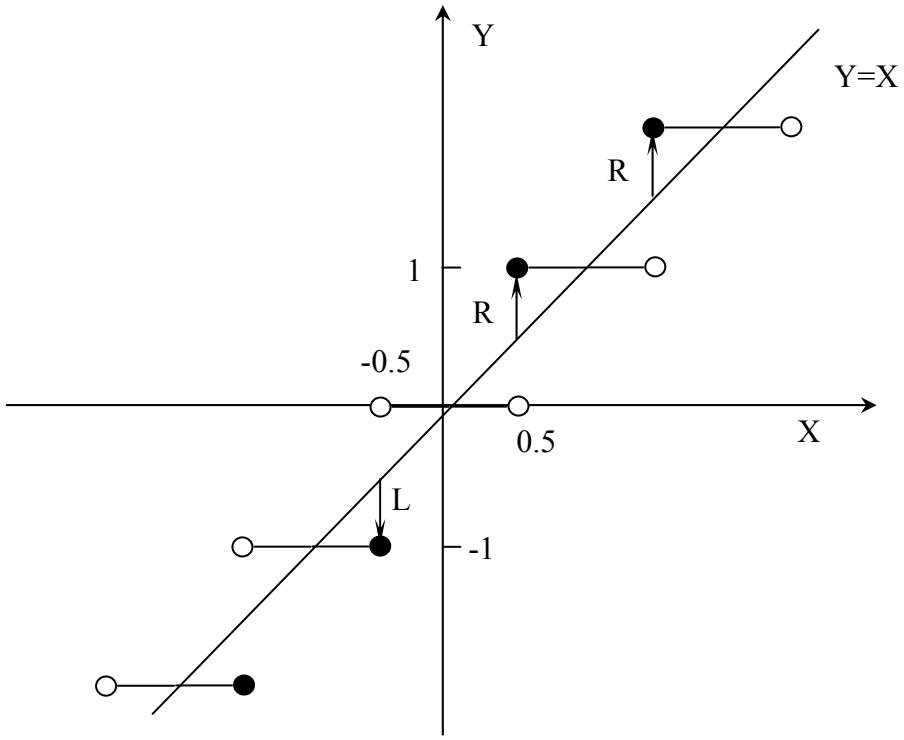

Fig 4.1. Rounding operation nr 3 – condition $\text{round}(x + int) = int + \text{round}(x)$ *is not always satisfied*

Among the above definitions of rounding only the first two have the latter property and this is why they can be used in the reasoning conducted here. The definition No 1 is chosen because for positive numbers it coincides with the definition No 3. The property **(4.03)** is explicitly exploited in Chapter 4 when deriving the equation **(4.54)** and **(4.73)**.

4.2 Linear transformation – definitions necessary for multiple encoding decoding cycles analysis

Let us consider Fig. 4.6 to address the problem of errors accumulation while the transform and the inverse transform are repeatedly performed.

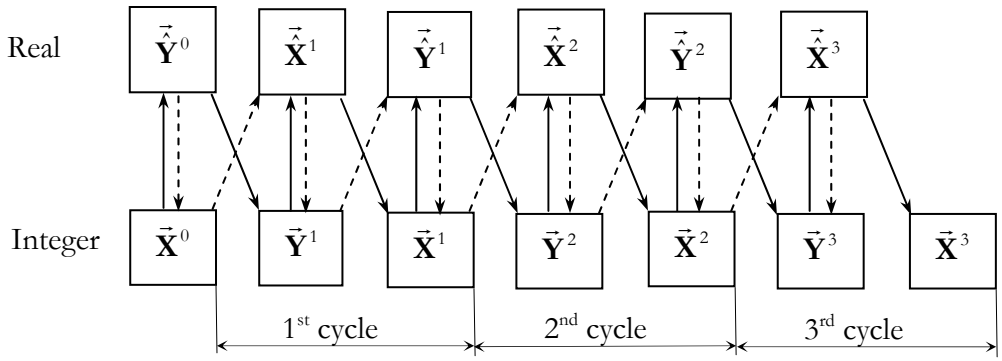

Figure 4.7. Consecutive encoding/decoding cycles. Exemplary interpretation of notation is given in Table 4.1

TABLE 4.1. EXEMPLARY INTEPRETATION OF NOTATION

General	Example 1	Example 2
$\vec{\mathbf{X}}$	R, G, B	8×8 luminance samples
$\vec{\mathbf{Y}}$	Y, C_B, C_R	8×8 DCT coefficients

In order to make the consideration clear the following notation and definitions are introduced:

- the original data vector indices are equal to zero (the vector itself and its components are not dashed);
- integer data vectors are not dashed;
- real data vectors are dashed;
- the error values are always real;

- the non-dashed errors (dashed) are added to the data components being also non-dashed (dashed);
- vertical arrows represent transformations;
- inclined arrows represent adding errors (in case they originate while either rounding or range clipping);
- solid arrows represent the operations being actually done while consecutive encoding/decoding cycles are performed (transformations and rounding);
- dashed arrows represent transformations and adding errors (implied by rounding), which can be derived analytically.

A linear transformation and the corresponding inverse transformation can be expressed as

$$\hat{\vec{\mathbf{Y}}}^{n} = \mathbf{T} \cdot \vec{\mathbf{X}}^{n} \quad (4.4) \qquad \vec{\mathbf{Y}}^{n} = \mathbf{T} \cdot \hat{\vec{\mathbf{X}}}^{n} \quad (4.5)$$

$$\vec{\mathbf{X}}^{n} = \mathbf{S} \cdot \hat{\vec{\mathbf{Y}}}^{n} \quad (4.6) \qquad \hat{\vec{\mathbf{X}}}^{n} = \mathbf{S} \cdot \vec{\mathbf{Y}}^{n}. \quad (4.7)$$

The superscripts in equations **(4.4)**, **(4.5)**, **(4.6)** and **(4.7)** denote the cycle number in multiple encoding and decoding.

The formulae presented below constitute the mathematical description of the operations graphically illustrated on Fig. 4.2:

$$\vec{\mathbf{Y}}^{n+1} = \text{round}\left(\hat{\vec{\mathbf{Y}}}^{n} \right), \quad (4.8)$$

as it is assumed that no range clipping occurs during the forward transform,

$$\vec{\mathbf{Y}}^{n+1} = \hat{\vec{\mathbf{Y}}}^{n} + \delta^{n} \hat{\vec{\mathbf{Y}}}, \quad (4.9)$$

$$\vec{\mathbf{X}}^{n} = \text{round } \textit{and / or } \text{clip}\left(\hat{\vec{\mathbf{X}}}^{n} \right), \quad (4.10)$$

and/or in the equation **(4.10)** means that some components can be rounded, while the others can be clipped,

$$\vec{\mathbf{X}}^{n} = \vec{\hat{\mathbf{X}}}^{n} + \delta^{n}\vec{\hat{\mathbf{X}}}. \tag{4.11}$$

The above equations are represented by solid inclined arrows in Fig. 4.2.

$$\vec{\hat{\mathbf{Y}}}^{n} = \vec{\mathbf{Y}}^{n} + \delta^{n}\vec{\mathbf{Y}}, \tag{4.12}$$

$$\vec{\hat{\mathbf{X}}}^{n+1} = \vec{\mathbf{X}}^{n} + \delta^{n}\vec{\mathbf{X}}. \tag{4.13}$$

The above equations are represented by dashed inclined arrows in Fig. 4.2.

$$\vec{\hat{\mathbf{Y}}}^{n} = \mathbf{T}\cdot\vec{\mathbf{X}}^{n}, \tag{4.14}$$

$$\vec{\hat{\mathbf{X}}}^{n} = \mathbf{T}^{-1}\cdot\vec{\mathbf{Y}}^{n}. \tag{4.15}$$

The above equations are represented by solid vertical arrows in Fig. 4.2.

$$\vec{\mathbf{Y}}^{n} = \mathbf{T}\cdot\vec{\hat{\mathbf{X}}}^{n}, \tag{4.16}$$

$$\vec{\mathbf{X}}^{n} = \mathbf{T}^{-1}\cdot\vec{\hat{\mathbf{Y}}}^{n}. \tag{4.17}$$

The above equations are represented by dashed vertical arrows in Fig. 4.2.

$$\delta^{n}\vec{\hat{\mathbf{Y}}} = \mathbf{T}\cdot\delta^{n}\vec{\mathbf{X}}, \tag{4.18}$$

$$\delta^{n}\vec{\mathbf{Y}} = \mathbf{T}\cdot\delta^{n}\vec{\hat{\mathbf{X}}}. \tag{4.19}$$

The above equations describe the dependencies between the operations represented by the arrows which cross each other (to be derived below). Every equation is valid at every second crossing on Fig. 4.2. To prove the first dependency **(4.18)** one should consider the transformation **(4.17)** for indices n+1:

$$\vec{\hat{\mathbf{X}}}^{n+1} = \mathbf{T}^{-1}\cdot\vec{\mathbf{Y}}^{n+1}. \tag{4.20}$$

Substituting equations **(4.13)** and **(4.9)** into **(4.20)** we get:

$$\vec{\mathbf{X}}^{n} + \delta^{n}\vec{\mathbf{X}} = \mathbf{T}^{-1}\cdot\left(\vec{\hat{\mathbf{Y}}}^{n} + \delta^{n}\vec{\mathbf{Y}}\right). \tag{4.21}$$

Equations **(4.17)** and **(4.21)** give:

$$\delta''\vec{\mathbf{X}} = \mathbf{T}^{-1} \cdot \delta''\hat{\vec{\mathbf{Y}}} . \qquad (4.22)$$

One can derive the second dependency **(4.19)** in a similar way.

4.3 Linear transformation – necessary condition for its reversibility

To map the set of vectors consisting of integer components into another set of vectors, also consisting of an integer components, with an arbitrary linear transform some rounding operation has to be applied. Such a transform can be considered **lossless** or **reversible** only if the superposition of the forward transform, followed by rounding and the inverse transform, followed by rounding is the identity transformation. The transformations which can be represented by the square, nonsingular, N – dimensional matrices are to be considered. A linear transform and the corresponding inverse one can be expressed as:

$$\vec{\mathbf{Y}} = \mathbf{T} \cdot \vec{\mathbf{X}} , \qquad (4.23)$$

$$\vec{\mathbf{X}} = \mathbf{S} \cdot \vec{\mathbf{Y}} . \qquad (4.24)$$

Following transformation matrix notation is introduced:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & & t_{1N} \\ t_{21} & t_{22} & & \\ & & & \\ t_{N1} & & & t_{NN} \end{bmatrix}, \qquad (4.25) \qquad \mathbf{S} = \mathbf{T}^{-1} = \begin{bmatrix} s_{11} & s_{12} & & s_{1N} \\ s_{21} & s_{22} & & \\ & & & \\ s_{N1} & & & s_{NN} \end{bmatrix} . \qquad (4.26)$$

It is convenient to introduce another symbols for following matrices:

$$\mathbf{T}_{abs} = \begin{bmatrix} |t_{11}| & |t_{12}| & & |t_{1N}| \\ |t_{21}| & |t_{22}| & & \\ & & & \\ |t_{N1}| & & & |t_{NN}| \end{bmatrix}, \qquad (4.27) \qquad \mathbf{S}_{abs} = \left(\mathbf{T}^{-1}\right)_{abs} = \begin{bmatrix} |s_{11}| & |s_{12}| & & |s_{1N}| \\ |s_{21}| & |s_{22}| & & \\ & & & \\ |s_{N1}| & & & |s_{NN}| \end{bmatrix} . \qquad (4.28)$$

Using fixed point representation of real numbers for storing data results in two effects:

1. discretisation,

2. dynamic range limitation.

The first effect can be imagined as selecting the countably infinite number of points in a continuous data space. The second one results in further reducing the number of points to the finite number of data points.

Assumptions

Data samples are points in an N – dimensional ($\mathbf{R}^N$) linear space $\boldsymbol{V}$ over the field $\mathbf{R}$ with a scalar product defined in it. Let the original set of base vectors be the ortho-normal set.

Proposition 1

The transform followed by rounding can be done losslessly (reversibly) only under the following necessary (but not sufficient) condition:

$$\left|\det(\mathbf{T})\right| \geq 1, \tag{4.29}$$

where $\mathbf{T}$ is a real transform what means that it can be represented by a matrix consisting of real elements.

Proof

The generalized volume (1D-length, 2D-area, 3D-volume) of the hyper-parallelepiped spanned by the set of N vectors $\overset{1}{\mathbf{v}}, \overset{2}{\mathbf{v}}, ..., \overset{N}{\mathbf{v}}$ is calculated in the following way:

$$vol(\overset{1}{\mathbf{v}}, \overset{2}{\mathbf{v}}, ..., \overset{N}{\mathbf{v}}) = \sqrt{\left|\det_G\left(\overset{1}{\mathbf{v}}, \overset{2}{\mathbf{v}}, ..., \overset{N}{\mathbf{v}}\right)\right|}, \tag{4.30}$$

where the Gram determinant $\det_G$ is defined as follows:

$$\det_G\left(\overset{1}{\mathbf{v}}, \overset{2}{\mathbf{v}}, ..., \overset{N}{\mathbf{v}}\right) = \det\begin{pmatrix} \overset{1}{\mathbf{v}}\cdot\overset{1}{\mathbf{v}} & \overset{1}{\mathbf{v}}\cdot\overset{2}{\mathbf{v}} & \overset{1}{\mathbf{v}}\cdot\overset{N}{\mathbf{v}} \\ \overset{2}{\mathbf{v}}\cdot\overset{1}{\mathbf{v}} & \overset{2}{\mathbf{v}}\cdot\overset{2}{\mathbf{v}} & \overset{2}{\mathbf{v}}\cdot\overset{N}{\mathbf{v}} \\ \\ \overset{N}{\mathbf{v}}\cdot\overset{1}{\mathbf{v}} & \overset{N}{\mathbf{v}}\cdot\overset{2}{\mathbf{v}} & \overset{N}{\mathbf{v}}\cdot\overset{N}{\mathbf{v}} \end{pmatrix}, \tag{4.31}$$

where the scalar product in the ortho-normal frame of reference is defined as follows:

$$\overset{k}{\mathbf{v}}\cdot\overset{l}{\mathbf{v}} = \sum_i \overset{k}{v}_i \cdot \overset{l}{v}_i, \tag{4.32}$$

where v_i^l is i'th component of l'th vector. Let us consider the linear transform $\mathbf{T}$ as a changing of frame of reference in $\boldsymbol{V}$. Let us recall the definitions **(4.05)** and **(4.06)**:

$$\mathbf{T} = \begin{bmatrix} t_{11} & t_{12} & t_{1N} \\ t_{21} & t_{22} & t_{2N} \\ & & \\ t_{N1} & t_{N2} & t_{NN} \end{bmatrix}, \quad \textbf{(4.05)} \qquad \mathbf{S} = \mathbf{T}^{-1} = \begin{bmatrix} s_{11} & s_{12} & s_{1N} \\ s_{21} & s_{22} & s_{2N} \\ & & \\ s_{N1} & s_{N2} & s_{NN} \end{bmatrix}. \quad \textbf{(4.06)}$$

In order to find out, what coordinates the new **base vectors** will have in the old frame of reference, let us consider the following equations:

$$\begin{bmatrix} s_{11} \\ s_{21} \\ \\ s_{N1} \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{1N} \\ s_{21} & s_{22} & s_{2N} \\ & & \\ s_{N1} & s_{N2} & s_{NN} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ \\ 0 \end{bmatrix}, \quad \begin{bmatrix} s_{12} \\ s_{22} \\ \\ s_{N2} \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{1N} \\ s_{21} & s_{22} & s_{2N} \\ & & \\ s_{N1} & s_{N2} & s_{NN} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ \\ 0 \end{bmatrix} \qquad \textbf{(4.33)}$$

and so on.

Thus every column of the $\mathbf{T}^{-1}$ matrix consists of the components of the new base vector represented in the old frame of reference. The **hyper-parallelepiped**, spanned on all the base vectors of the new frame of reference, is an **elementary cell** of the new frame of reference. Side faces of the parallelepiped, which a parallel to each other, cannot both belong to the elementary cell but only one such face belongs to it. One can reproduce this elementary cell to any countable number. It is possible to partition the data samples space between those new elementary cells uniquely, by shifting each of them by different linear combination of the elementary vectors. The coefficients of these linear combinations must be integer numbers. Those elementary cells are shifted in such a way, that they have the data points with integer coordinates (components) in their centers. Then all the data points, included in such a cell, will be mapped to the cell center if rounding is performed.

The Gram determinant for such an elementary cell can be computed as follows:

$$\det_G\left(\overset{1}{\mathbf{v}}, \overset{2}{\mathbf{v}}, \ldots, \overset{N}{\mathbf{v}}\right) = \det\left(\left(\mathbf{T}^{-1}\right)^T \cdot \left(\mathbf{T}^{-1}\right)\right) = \det\left(\mathbf{T}^{-1}\right)^T \cdot \det\left(\mathbf{T}^{-1}\right) = \left(\det\left(\mathbf{T}^{-1}\right)\right)^2. \qquad \textbf{(4.34)}$$

Thus the volume of the considered parallelepiped is:

$$vol(\overset{1}{\mathbf{v}},\overset{2}{\mathbf{v}},...,\overset{N}{\mathbf{v}}) = \sqrt{\left|\det_G\left(\overset{1}{\mathbf{v}},\overset{2}{\mathbf{v}},...,\overset{N}{\mathbf{v}}\right)\right|} = \left|\det\left(\mathbf{T}^{-1}\right)\right| = \left|\det\left(\mathbf{T}\right)\right|^{-1}. \qquad \textbf{(4.35)}$$

Volume of the elementary **hyper-cube** spanned on the ortho-normal set of the original base vectors (elementary cell of the old frame of reference – elementary cube) has the unit generalized volume. The ratio of the elementary parallelepiped volume and the elementary cube volume determines the expected number of vectors, having integer components in old frame of reference, mapped to one vector, having integer components in the new frame of reference. It is obvious, that the transform followed by rounding can be done losslessly only under the following necessary (but not sufficient) condition:

$$\left|\det\left(\mathbf{T}^{-1}\right)\right| \le 1 \Leftrightarrow \left|\det(\mathbf{T})\right| \ge 1 \qquad \textbf{(4.36)}$$

QED.

All the following figures follow the same scheme. The black dots represent the data samples having integer components in the old frame of reference. The crossed circles represent the data samples having integer components in the new frame of reference. The squares drawn with dotted lines represent the elementary cell in the old frame of reference. Squares and parallelograms drawn with the solid lines represent the elementary cells in the new frame of reference. The elementary cells of both the frames of reference are shifted relative the integer data samples of corresponding frames of reference in such a way, that every cell has its integer data sample located in its center.

Fig. 4.2 shows why the transform with $|\det[\mathbf{T}]|<1$ is lossy.

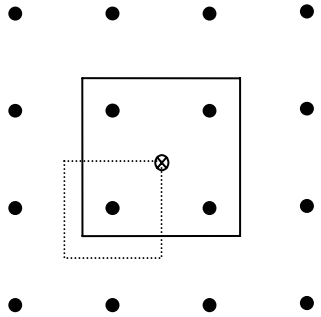

Fig 4.2. Lossy transform with $|\det[\mathbf{T}]|<1$.
solid line- elementary cells in the new frame of

On Fig. 4.2 four different black dots will be mapped to one crossed circle if rounding is performed. Fig. 4.3 shows why the transform with $|\det[\mathbf{T}]|=1$ can be lossless.

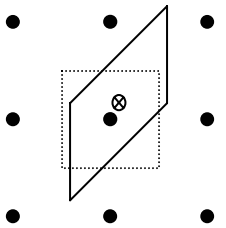

Fig 4.3. Lossless transform satisfying the condition: $|\det[\mathbf{T}]|=1$.
solid line - elementary cells in the new frame of reference

On Fig. 4.3 just one black dots will be mapped to one crossed circle if rounding is performed

Fig. 4.4 shows the case when $|\det[\mathbf{T}]|=1$ does not guarantee the transform reversibility.

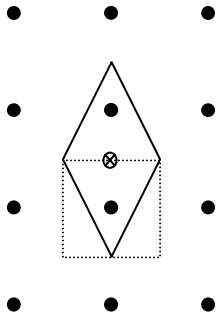

Fig 4.4. Lossy transform satisfying the condition: $|\det[\mathbf{T}]|=1$.
solid line - elementary cells in the new frame of reference

On Fig. 4.4 two different black dots will be mapped to one crossed circle if rounding is performed.

4.4 Linear transformation – sufficient condition for its reversibility and maximum rounding error in a single cycle for the irreversible one

Let us consider maximum errors which may be introduced into the data when transforming them and rounding the results, provided the very transform is a linear one to one mapping. Since the numbers are stored in the variables of a finite precision a transform may cause rounding and range clipping. The precision of variables is adjusted to the demanded dynamic range hence the transforms to be considered here do not cause range clipping when passing from the old one to some new frame of reference. This may happen, however, when the inverse transform is done. As mentioned above $t_{i,j}$ is the element of an arbitrarily chosen transform square matrix **T** and $s_{i,j}$ is the element of an **S** matrix, which is inverse to the latter one. Such a transform may be considered as a changing of a frame of reference in an appropriate space (e.g. color space, samples space). Both old and new frames of reference are provided to be rectilinear. It is assumed that data are represented by $\vec{\mathbf{X}}$ vector (it may consist three components for color representation, or 64 components for 8×8 block of luminance samples). The $\vec{\mathbf{X}}^0$, is the original vector consisting of integer component samples (cycle number 0). Then the $\vec{\mathbf{X}}^0$ vector components are transformed to a new system according to Equation **(4.4)**. The exact $\hat{\vec{\mathbf{Y}}}^0 = [\hat{Y}_1^0, \hat{Y}_2^0, \hat{Y}_3^0, ...]$ component values, obtained according to Eq. **(4.4)**, have to be rounded in order to obtain integers $\vec{\mathbf{Y}}^1 = [Y_1^1, Y_2^1, Y_3^1, ...]$.

Therefore there is

$$
\begin{aligned}
Y_1^1 &= \hat{Y}_1^0 + \delta^0 \hat{Y}_1 \ , \\
Y_2^1 &= \hat{Y}_2^0 + \delta^0 \hat{Y}_2 \ , \\
&\ldots, \\
Y_N^1 &= Y_N^0 + \delta^0 Y_N \ ,
\end{aligned}
\tag{4.37}
$$

where superscripts denote the encoding/decoding cycle number, and subscripts denote the component of the data vector.

There is:

$$
0 \le \left| \delta'' \hat{Y}_1 \right|, \left| \delta'' \hat{Y}_2 \right|, \ldots, \left| \delta'' \hat{Y}_N \right| \le 0.5 \, . \tag{4.38}
$$

The data vector $\vec{\mathbf{Y}}^1$ has to be converted back to the $\mathbf{X}$ system by use of Eq. **(4.7)**.

Then again the recovered $\hat{\vec{\mathbf{X}}}^1$ data vector obtained from Eq. **(4.7)** has to be rounded to integer data vector $\vec{\mathbf{X}}^1$ within an appropriate range. Let us denote:

$$
\begin{aligned}
\delta_1 &= X_1^1 - X_1^0 , \\
\delta_2 &= X_2^1 - X_2^0 , \\
&\ldots, \\
\delta_N &= X_N^1 - X_N^0 ,
\end{aligned}
\tag{4.39}
$$

Therefore $\delta_1, \delta_2, \ldots, \delta_N$, denote the differences between the input $X_1^0, X_2^0, \ldots, X_N^0$ sample values and those $X_1^1, X_2^1, \ldots, X_N^1$ rounded after inverse transformation. The values $\delta_1, \delta_2, \ldots, \delta_N$, are integer because $X_1^0, X_2^0, \ldots, X_N^0$ and $X_1^1, X_2^1, \ldots, X_N^1$ are also integer.

Proposition 2 (published in [Domanski1] and [Domanski2])

Sample differences $\delta_1, \delta_2, \ldots, \delta_N$, of the $X_1, X_2, \ldots, X_N$ components in the single cycle of forward and backward transformations are bounded $|\delta_1| \le Bnd_1, |\delta_2| \le Bnd_2, \ldots, |\delta_N| \le Bnd_N$, provided no range clipping occurs while either the transform or the inverse transform is done.

Where

$$
\begin{aligned}
Bnd_1 &= \text{round}[\delta X_1^{\max}], \\
Bnd_2 &= \text{round}[\delta X_2^{\max}], \\
&\ldots, \\
Bnd_N &= \text{round}[\delta X_N^{\max}],
\end{aligned}
\tag{4.40}
$$

for $\delta X_k^{\max} = \frac{1}{2} \cdot \sum_{i=1}^{N} \left| s_{k,i} \right|$.

Proof:

It is assumed that variables used for storing integer $\vec{\mathbf{Y}}$ coordinates have precision sufficient for the demanded dynamic range. The dynamic range demanded for the new $\vec{\tilde{\mathbf{Y}}}$ coordinates can be calculated by means of norm L_∞ of **T** matrix multiplied by maximum possible value of original sample ($X_1^0, X_2^0, \ldots$ or X_N^0 - provided all of them have the same dynamic range).

Equation **(4.7)** yields

$$
\begin{bmatrix} \hat{X}_1^1 \\ \hat{X}_2^1 \\ \\ \hat{X}_N^1 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & & s_{1N} \\ s_{21} & s_{22} & & \\ & & & \\ s_{N1} & & & s_{NN} \end{bmatrix} \begin{bmatrix} Y_1^1 \\ Y_2^1 \\ \\ Y_N^1 \end{bmatrix}. \tag{4.41}
$$

When substituting equation nr **(4.37)** into the above one we get:

$$
\begin{bmatrix} \hat{X}_1^1 \\ \hat{X}_2^1 \\ \\ \hat{X}_N^1 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & & s_{1N} \\ s_{21} & s_{22} & & \\ & & & \\ s_{N1} & & & s_{NN} \end{bmatrix} \begin{bmatrix} \hat{Y}_1^0 + \delta^0 \hat{Y}_1 \\ \hat{Y}_2^0 + \delta^0 \hat{Y}_2 \\ \\ \hat{Y}_N^0 + \delta^0 \hat{Y}_N \end{bmatrix}. \tag{4.42}
$$

Equation **(4.6)** yields

$$
\begin{bmatrix} X_1^0 \\ X_2^0 \\ \\ X_N^0 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & & s_{1N} \\ s_{21} & s_{22} & & \\ & & & \\ s_{N1} & & & s_{NN} \end{bmatrix} \begin{bmatrix} \hat{Y}_1^0 \\ \hat{Y}_2^0 \\ \\ \hat{Y}_N^0 \end{bmatrix}. \tag{4.43}
$$

Subtracting the above equation from equation **(4.42)** we get:

$$\begin{bmatrix} \hat{X}_1^1 - X_1^0 \\ \hat{X}_2^1 - X_2^0 \\ \\ \hat{X}_N^1 - X_N^0 \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & & s_{1N} \\ s_{21} & s_{22} & & \\ & & & \\ s_{N1} & & & s_{NN} \end{bmatrix} \begin{bmatrix} \delta^0 \hat{Y}_1 \\ \delta^0 \hat{Y}_2 \\ \\ \delta^0 \hat{Y}_N \end{bmatrix}. \qquad \textbf{(4.44)}$$

Hence:

$$\begin{aligned} \left|\hat{X}_1^1 - X_1^0\right| &\le \left|s_{11} \cdot \delta^0 \hat{Y}_1\right| + \left|s_{12} \cdot \delta^0 \hat{Y}_2\right| + ... + \left|s_{1N} \cdot \delta_0{}^0 \hat{Y}_3\right|, \\ \left|\hat{X}_2^1 - X_2^0\right| &\le \left|s_{21} \cdot \delta^0 \hat{Y}_1\right| + \left|s_{22} \cdot \delta^0 \hat{Y}_2\right| + ... + \left|s_{2N} \cdot \delta^0 \hat{Y}_3\right|, \\ &..., \\ \left|\hat{X}_N^1 - X_N^0\right| &\le \left|s_{N1} \cdot \delta^0 \hat{Y}_1\right| + \left|s_{N2} \cdot \delta^0 \hat{Y}_2\right| + ... + \left|s_{NN} \cdot \delta^0 \hat{Y}_3\right|. \end{aligned} \qquad \textbf{(4.45)}$$

Relation $0 \le \left|\delta'' \hat{Y}_1\right|, \left|\delta'' \hat{Y}_2\right|, ..., \left|\delta'' \hat{Y}_N\right| \le 0.5$ (variables have sufficient precision to represent all the dynamic range of **Y** components) implies

$$\begin{aligned} \left|\hat{X}_1^1 - X_1^0\right| &\le 0.5 \cdot \left(\left|s_{11}\right| + \left|s_{12}\right| + ... + \left|s_{1N}\right|\right), \\ \left|\hat{X}_2^1 - X_2^0\right| &\le 0.5 \cdot \left(\left|s_{21}\right| + \left|s_{22}\right| + ... + \left|s_{2N}\right|\right), \\ &..., \\ \left|\hat{X}_N^1 - X_N^0\right| &\le 0.5 \cdot \left(\left|s_{N1}\right| + \left|s_{N2}\right| + ... + \left|s_{NN}\right|\right), \end{aligned} \qquad \textbf{(4.46)}$$

The values of $X_1^0, X_2^0, ...$ and X_N^0 are integer, therefore rounding of $\hat{X}_1^1, \hat{X}_2^1, ..., \hat{X}_N^1$ to the nearest integer (in general $\hat{X}_1^1, \hat{X}_2^1, ..., \hat{X}_N^1$ may be rounded not to the nearest integer because of range clipping, it is assumed range clipping occurs neither here) will result in values $X_1^1, X_2^1, ..., X_N^1$ such that

$$\begin{aligned} \left|\delta_1\right| &= \left|X_1^1 - X_1^0\right| \le \text{round}((|s_{11}| + |s_{12}| + ... + |s_{1N}|)/2), \\ \left|\delta_2\right| &= \left|X_2^1 - X_2^0\right| \le \text{round}((|s_{21}| + |s_{22}| + ... + |s_{2N}|)/2), \\ &..., \\ \left|\delta_N\right| &= \left|X_N^1 - X_N^0\right| \le \text{round}((|s_{N1}| + |s_{N2}| + ... + |s_{NN}|)/2). \end{aligned} \qquad \textbf{(4.47)}$$

Q.E.D.

As the rounding errors are limited by the above relations it is possible to find out when such a transform combined with rounding is lossless. It is enough to check if the right hand side of each of the above inequalities comes to zero after rounding. This is so when the norm L_∞ of the

matrix **S** is smaller then 1, provided the variables used for computations have precision sufficient for a demanded dynamic range (no range clipping occurs when transformation **T** is done). The above mentioned conclusions can be explicitly expressed as it is done below:

$$\begin{aligned}&(|s_{11}|+|s_{12}|+...+|s_{1N}|)<1,\\&(|s_{21}|+|s_{22}|+...+|s_{2N}|)<1,\\&...,\\&(|s_{N1}|+|s_{N2}|+...+|s_{NN}|)<1,\end{aligned} \qquad \textbf{(4.48)}$$

or

$$L_\infty(\mathbf{T}^{-1})<1. \qquad \textbf{(4.49)}$$

Hence if the inverse transform reduces the dynamic range of the data, the forwards transform followed by rounding will be entirely lossless. Hence, Equation **(4.49)** gives the sufficient condition for the transformation reversibility. It is easy to prove that if the sufficient condition **(4.49)** is satisfied, then the necessary condition **(4.29)** will be satisfied. If the transform **T** combined with rounding is lossy (norm L_∞ of matrix $\mathbf{S}=\mathbf{T}^{-1}$ is not smaller then one) it will be possible to modify it by multiplying **T** by a number grater then L_∞ of the **S** matrix getting an entirely lossless transform **T'**. In such a case variables used for storing the new **Y** coordinates must be given higher precision (to match the increased dynamic range) and matrix **S** must by divided by the same number so it will be inverse to the new **T'** lossless transform.

4.5 Conditions for error accumulation termination

Let us define **error saturation**.

Error saturation is achieved in a certain cycle when all the subsequent encoding/decoding cycles introduce no further errors into the data.

Another class of linear transform is very likely to introduce rounding errors into the data only in the first encoding decoding cycle. Such transforms have the following property:

Proposition 3 (published in [Domanski1] and [Domanski2])

Providing no range clipping occurs while the forwards transform is done the conditions:

$$L_\infty(\mathbf{T}) < 1 \tag{4.50}$$

yield the following property:

after the cycle, in which the range clipping has not occurred during the inverse transform, all the consecutive cycles of forward and backward transformations introduce no further errors into the data, hence error saturation is achieved.

Remarks.

The conditions used in the above proposition are very similar to those which guarantee that transformed data dynamic range will not exceed the dynamic range of not transformed data

$$L_\infty(\mathbf{T}) \le 1. \tag{4.51}$$

This means all the transforms which reduce the dynamic range of transformed data have the above property. Both above mentioned sets of conditions can be expressed by means of infinity norm L_∞ of the $\mathbf{T}$ matrix. It is enough to demand that L_∞ norm of $\mathbf{T}$ is smaller (or not greater) then 1.

Proof:

Equations **(4.8)** and **(4.12)** give:

$$\vec{\mathbf{Y}}^{n+1} = \text{round}\left(\hat{\vec{\mathbf{Y}}}^{n}\right), \tag{4.52}$$

$$\hat{\vec{\mathbf{Y}}}^{n} = \vec{\mathbf{Y}}^{n} + \delta^{n}\vec{\mathbf{Y}}. \tag{4.53}$$

Substituting **(4.53)** into **(4.52)** results in:

$$\vec{\mathbf{Y}}^{n+1} = \text{round}\left(\vec{\mathbf{Y}}^{n} + \delta^{n}\vec{\mathbf{Y}}\right). \tag{4.54}$$

Since vector $\vec{\mathbf{Y}}^{n}$ consists of integer numbers one can write:

$$\vec{\mathbf{Y}}^{n+1} = \vec{\mathbf{Y}}^{n} + \text{round}\left(\delta^{n}\vec{\mathbf{Y}}\right). \tag{4.55}$$

Substituting **(4.19)** into **(4.55)** one gets:

$$\vec{\mathbf{Y}}^{n+1} = \vec{\mathbf{Y}}^{n} + \text{round}\left(\mathbf{T}\cdot\delta^{n}\hat{\vec{\mathbf{X}}}\right). \tag{4.56}$$

Equation **(4.56)** implies:

$$\vec{\mathbf{Y}}^{n+1} - \vec{\mathbf{Y}}^{n} = \mathrm{round}\left(\mathbf{T} \cdot \delta^{n}\vec{\hat{\mathbf{X}}}\right). \qquad \textbf{(4.57)}$$

To find out if the error saturation is achieved in an *n*'th cycle, one should take equation **(4.57)** into account.

$$\text{If } \mathrm{round}\left(\mathbf{T} \cdot \delta^{n}\vec{\hat{\mathbf{X}}}\right) = \vec{\mathbf{0}} \qquad \textbf{(4.58)}$$

$$\text{for all possible } \delta^{n}\vec{\hat{\mathbf{X}}}, \qquad \textbf{(4.59)}$$

$$\text{then} \quad \vec{\mathbf{Y}}^{n+1} = \vec{\mathbf{Y}}^{n} \text{ thus } \vec{\mathbf{X}}^{n+1} = \vec{\mathbf{X}}^{n} \qquad \textbf{(4.60)}$$

and the second encoding decoding cycle does not introduce any additional errors into the data.

The absolute values of dashed **X** errors may be greater then half only because of range clipping. Hence error saturation will be achieved in *n*'th cycle if there is no range clipping during the *n*'th inverse transform $(|\delta^{n}\hat{X}_1|, |\delta^{n}\hat{X}_2|, ..., |\delta^{n}\hat{X}_N| \le 0.5)$ and if the following conditions are satisfied

$$\begin{bmatrix} |t_{11}| & |t_{12}| & & |t_{1N}| \\ |t_{21}| & |t_{22}| & & \\ & & & \\ |t_{N1}| & & & |t_{NN}| \end{bmatrix} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ \\ 0.5 \end{bmatrix} < \begin{bmatrix} 0.5 \\ 0.5 \\ \\ 0.5 \end{bmatrix}. \qquad \textbf{(4.61)}$$

Hence:

$$\begin{aligned} &(|t_{11}| + |t_{12}| + ... + |t_{1N}|) < 1, \\ &(|t_{21}| + |t_{22}| + ... + |t_{2N}|) < 1, \\ &..., \\ &(|t_{N1}| + |t_{N2}| + ... + |t_{NN}|) < 1. \end{aligned} \qquad \textbf{(4.62)}$$

The above conditions are equivalent to the following one:

$$L_{\infty}(\mathbf{T}) < 1. \qquad \textbf{(4.63)}$$

Q.E.D.

4.6 Error accumulation in multiple encoding/decoding cycles

Summarizing the two previous sections one can say:

- Transformation followed by rounding is losselss if $L_\infty(\mathbf{T}^{-1}) < 1$.
- If the range clipping is impossible during the forward transform and does not occur in the inverse transform of the n-th cycle, and if $L_\infty(\mathbf{T}) < 1$ then after the n-th encoding/decoding cycle no additional errors are introduced into the data.

Let us analyze the case when $L_\infty(\mathbf{T}) < 1$ and the range clipping is impossible after the forwards transform but may occur after the inverse transform. There is no need to consider the range clipping during a forward transform because the precision of variables, used for storing the transformed data, always match the dynamic range.

4.6.1 Recursive formulae for an error evolution

In order to understand the process of transforming the data forwards and backwards let us derive recursive formulae. For the beginning, the lack of range clipping is assumed.

Equations **(4.13)** and **(4.11)** imply:

$$\begin{aligned} &\vec{\mathbf{X}}^{n+1} - \hat{\vec{\mathbf{X}}}^{n} = \\ &= \vec{\mathbf{X}}^{n} + \delta^{n}\vec{\mathbf{X}} - \left(\vec{\mathbf{X}}^{n} - \delta^{n}\hat{\vec{\mathbf{X}}}\right) = \\ &= \delta^{n}\vec{\mathbf{X}} + \delta^{n}\hat{\vec{\mathbf{X}}} \end{aligned} \qquad \textbf{(4.64)}$$

for $n = 1, 2, \ldots N$.

Substituting equation **(4.16)** into **(4.57)** we get:

$$\mathbf{T}\cdot\hat{\vec{\mathbf{X}}}^{n+1} - \mathbf{T}\cdot\hat{\vec{\mathbf{X}}}^{n} = \mathrm{round}\left(\mathbf{T}\cdot\delta^{n}\hat{\vec{\mathbf{X}}}\right) \qquad \textbf{(4.65)}$$

$$\text{and } \hat{\vec{\mathbf{X}}}^{n+1} - \hat{\vec{\mathbf{X}}}^{n} = \mathbf{T}^{-1}\cdot\mathrm{round}\left(\mathbf{T}\cdot\delta^{n}\hat{\vec{\mathbf{X}}}\right). \qquad \textbf{(4.66)}$$

After substituting **(4.64)** into **(4.66)** we obtain:

$$\delta^n\vec{\mathbf{X}} + \delta^n\vec{\hat{\mathbf{X}}} = \mathbf{T}^{-1}\cdot\mathrm{round}\left(\mathbf{T}\cdot\ \delta^n\vec{\hat{\mathbf{X}}}\right). \qquad (4.67)$$

Thus we get an important result:

$$\delta^n\vec{\mathbf{X}} = -\delta^n\vec{\hat{\mathbf{X}}} + \mathbf{T}^{-1}\cdot\mathrm{round}\left(\mathbf{T}\cdot\ \delta^n\vec{\hat{\mathbf{X}}}\right). \qquad (4.68)$$

The last constituent in the equation **(4.68)** can be called a **secondary migration vector** in **X** frame of reference:

$$\delta^n\vec{\hat{\mathbf{X}}}^{Mig2} = \mathbf{T}^{-1}\cdot\mathrm{round}\left(\mathbf{T}\cdot\ \delta^n\vec{\hat{\mathbf{X}}}\right). \qquad (4.69)$$

The same vector in the **Y** frame of reference is as follows:

$$\delta^n\vec{\mathbf{Y}}^{Mig2} = \mathrm{round}\left(\mathbf{T}\cdot\ \delta^n\vec{\hat{\mathbf{X}}}\right). \qquad (4.70)$$

These migration vectors represent the migration between the Y elementary cell centers. Accordingly such a vector both stats and ends in a point having integer Y coordinates.

Let us rewrite the equation **(4.10)** for the cycle n+1, providing no range clipping occurs

$$\vec{\mathbf{X}}^{n+1} = \mathrm{round}\left(\vec{\hat{\mathbf{X}}}^{n+1}\right). \qquad (4.71)$$

Hence:

$$\vec{\mathbf{X}}^{n+1} = \vec{\mathbf{X}}^{n} - \vec{\mathbf{X}}^{n} + \mathrm{round}\left(\vec{\hat{\mathbf{X}}}^{n+1}\right). \qquad (4.72)$$

Since vector $\vec{\mathbf{X}}^n$ consists of integers we can write:

$$\vec{\mathbf{X}}^{n+1} = \vec{\mathbf{X}}^{n} + \mathrm{round}\left(\vec{\hat{\mathbf{X}}}^{n+1} - \vec{\mathbf{X}}^{n}\right). \qquad (4.73)$$

Let us recall the definition **(4.13)**

$$\vec{\hat{\mathbf{X}}}^{n+1} = \vec{\mathbf{X}}^{n} + \delta^n\vec{\mathbf{X}}. \qquad (4.74)$$

Equation **(4.74)** implies

$$\delta^n\vec{\mathbf{X}} = \vec{\hat{\mathbf{X}}}^{n+1} - \vec{\mathbf{X}}^{n}. \qquad (4.75)$$

After substituting **(4.75)** into **(4.73)** we get:

$$\vec{\mathbf{X}}^{n+1} = \vec{\mathbf{X}}^{n} + \text{round}\left(\delta^{n}\vec{\mathbf{X}}\right). \tag{4.76}$$

The definition **(4.11)** implies:

$$\delta^{n+1}\vec{\hat{\mathbf{X}}} = \vec{\mathbf{X}}^{n+1} - \vec{\hat{\mathbf{X}}}^{n+1}. \tag{4.77}$$

Substituting **(4.76)** into **(4.77)** we get:

$$\delta^{n+1}\vec{\hat{\mathbf{X}}} = \vec{\mathbf{X}}^{n} + \text{round}\left(\delta^{n}\vec{\mathbf{X}}\right) - \vec{\hat{\mathbf{X}}}^{n+1}. \tag{4.78}$$

Using the definition (4.13) we can write the equation (4.78) as follows:

$$\delta^{n+1}\vec{\hat{\mathbf{X}}} = -\delta^{n}\vec{\mathbf{X}} + \text{round}\left(\delta^{n}\vec{\mathbf{X}}\right). \tag{4.79}$$

The last constituent in the equation **(4.79)** can be called a primary migration vector in $\mathbf{X}$ frame of reference:

$$\delta^{n}\vec{\mathbf{X}}^{Mig1} = \text{round}\left(\delta^{n}\vec{\mathbf{X}}\right). \tag{4.80}$$

Such a migration vector represents the migration between the X elementary cell centers. When comparing the Definition **(4.80)** with Definitions **(4.81)** one can see, that the proposition 2 from Section 4.4 defines the upper bounds for the absolute values of primary migration vector components.

The set of recursive formulas (**(4.68)** and **(4.79)**) has been obtained. They describe the dependencies between the rounding errors which originate while the consecutive encoding/decoding cycles are done. Fig. 4.6 illustrates the Equation 4.77 in two dimensions for the case when no range clipping occurs.

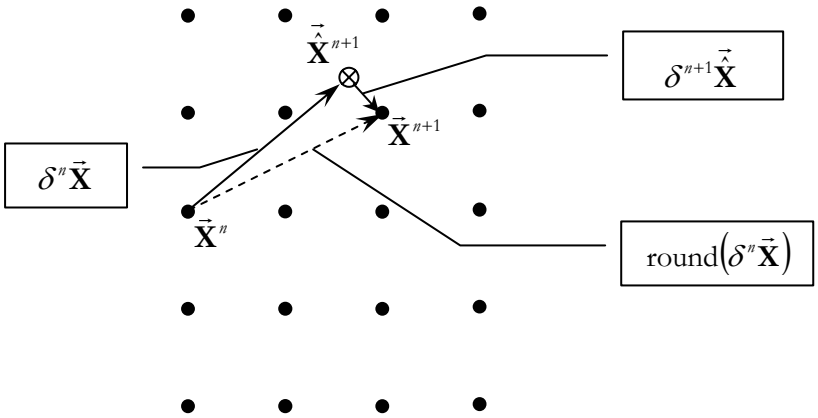

Fig 4.6. The illustration of the equation 4.79:
$\delta^{n+1}\hat{\vec{\mathbf{X}}} = -\delta^{n}\vec{\mathbf{X}} + \mathrm{round}\left(\delta^{n}\vec{\mathbf{X}}\right)$

The black dots represent the data samples having integer values in the X frame of reference. The empty crossed circle represents one of the data samples having integer values in Y frame of reference (and possibly having real values in X frame of reference). The arrows represent the increments taken from the equation 4.79. The arrow drawn with a dashed line (primary migration vector) represents the data point migration, which is due to rounding in single encoding/decoding cycle. The accumulated primary migration vectors for a few consecutive encoding/decoding cycles show, how far away from its original value the data point has migrated. Similar figure could be drawn for the Eq. 4.68. However, the dashed arrow (secondary migration vector) would start and end in the crossed circle in such a figure.

4.6.2 Range clipping

The formulae (4.68) and (4.79) are valid under the assumption that no range clipping occurs. If range clipping is impossible during the forward transform, but allowed during the inverse one, then the equation (4.79) will have to be modified.

To approach this problem let us redraw Fig. 4.6, which is not valid for range clipping, adding the axes X_1 and X_2.

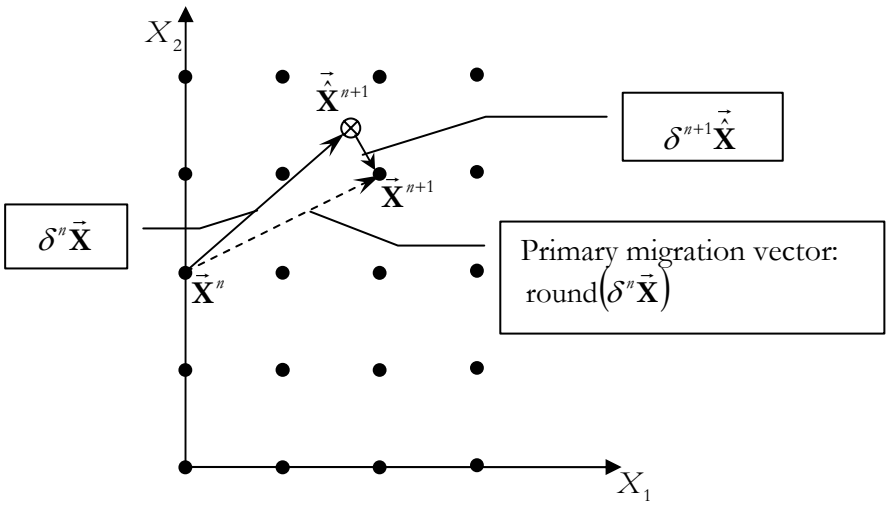

Fig 4.7. The two-dimensional case of equation 4.79.
Both the components are rounded:

$$\begin{bmatrix}\delta^{n+1}\hat{X}_1\\ \delta^{n+1}\hat{X}_2\end{bmatrix}=-\begin{bmatrix}\delta^{n}X_1\\ \delta^{n}X_2\end{bmatrix}+\begin{bmatrix}\text{round}\left(\delta^{n}X_1\right)\\ \text{round}\left(\delta^{n}X_2\right)\end{bmatrix}$$

In the subtitle of Fig. 4.7 the two dimensional version of the equation (4.79) is shown. Every vector drawn on this figure consists of two components. Those components are explicitly shown in the formula, which is presented in the figure subtitle.

In order to analyze the range clipping both the above figure and equation will have to be further modified. Let us define an **X range hyper-cuboid** as a cuboid being a subset of a data space within a dynamic range of an input data. In a three dimensional space a three dimensional range cuboid and in an N – dimensional space an N – dimensional range hyper-cuboid must be considered. In two dimensions such a range hyper-cuboid turns into a rectangle. More dimensions are considered, more kinds of hyper-edges such a hyper-cuboid has, which altogether constitute its borders. The X range hyper-cuboid without its borders is an open set. The hyper-edges of the X range hyper-cuboid with dimensionality (N-M) are also the borders of certain (N-M+1) – dimensional hyper-edge of the range hyper-cuboid. For example, a three-dimensional range hyper-cuboid has eight zero dimensional hyper-edges called "corners", twelve one dimensional hyper-edges called "edges" and six two dimensional hyper-edges called "side faces". Hence when clipping three components of the data point a zero dimensional hyper-edge is achieved (corner), when clipping two components of the data point a one dimensional hyper-

edge is achieved (edge) and when clipping one component of the data point a two dimensional hyper-edge is achieved (side face). Accordingly, in an N – dimensional data space, the clipping of M components of data point results in achieving the (N-M)– dimensional hyper-edge. From now on the word "corner" will refer to the zero dimensional hyper-edge of the N – dimensional range hyper-cuboid, and the word "side face" will refer to the two dimensional hyper-edge of the N – dimensional range hyper-cuboid.

The question to be discussed is how the range clipping, which may happen when the inverse transform is done, affects Proposition 2 (refer to Section 4.4), which is treating about maximum errors due to single encoding/decoding cycle.

Let us redraw Fig. 4.7 modifying it to show the range clipping phenomenon for the $\vec{\mathbf{X}}^n$ belonging to the edge of the two dimensional X range cuboid.

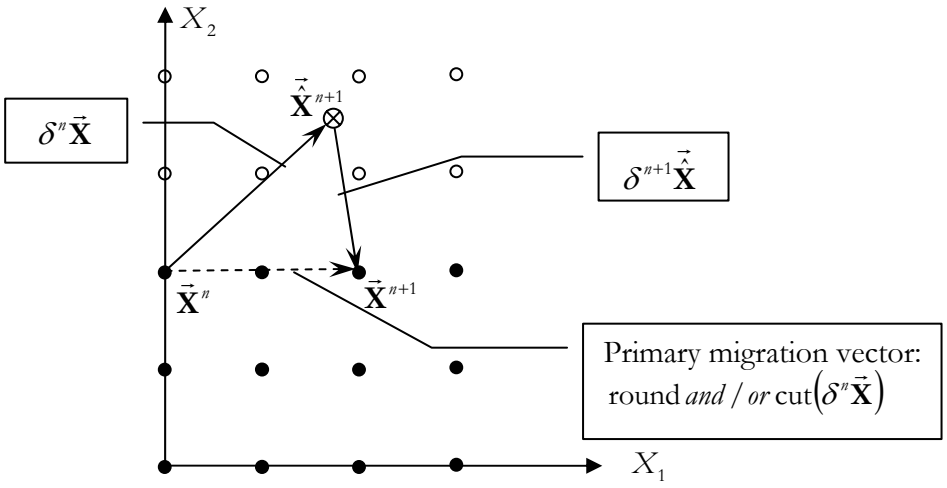

Fig 4.8. The two-dimensional case of equation 4.79. The first component X_1 is rounded and the second one X_2 is clipped:

$$\begin{bmatrix}\delta^{n+1}\hat{X}_1\\ \delta^{n+1}\hat{X}_2\end{bmatrix}=-\begin{bmatrix}\delta^{n}X_1\\ \delta^{n}X_2\end{bmatrix}+\begin{bmatrix}\text{round}\left(\delta^{n}X_1\right)\\ 0\end{bmatrix}$$

Fig. 4.8 shows the range clipping for this situation. Empty small circles represent those data points which have integer components in X frame of references, lying outside of the X range cuboid. Black dots represent those data points which have integer components in X frame of

reference, belonging to the X range cuboid. One can see that a data point migrates, as the primary migration vector shows, along the edge of the X range cuboid.

Hence it becomes clear that in this case in equation **(4.79)**, for the component which is clipped, the rounding function must be canceled and zero must be put instead. This operation will be called "**cutting**". The cutting of the increments is equivalent to clipping of components.

Let us redraw Fig. 4.8 modifying it to show the range clipping phenomenon for the $\vec{\mathbf{X}}^n$ belonging to the interior of the two-dimensional range cuboid and one unit distant from its edge.

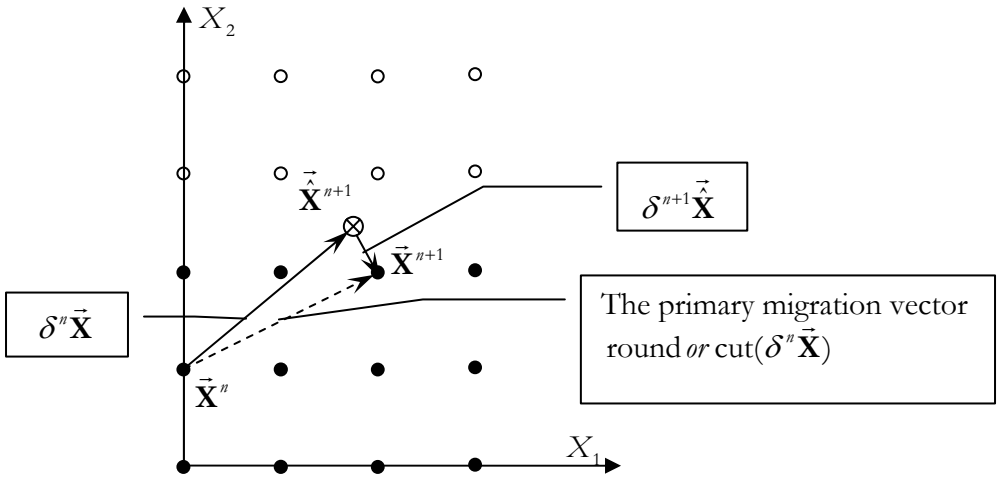

Fig. 4.9 The two-dimensional case of equation 4.79. The first component X_1 is Rounded and the second one X_2 is clipped:

$$\begin{bmatrix} \delta^{n+1}\hat{X}_1 \\ \delta^{n+1}\hat{X}_2 \end{bmatrix} = -\begin{bmatrix} \delta^n X_1 \\ \delta^n X_2 \end{bmatrix} + \begin{bmatrix} \text{round}(\delta^n X_1) \\ \text{sign}(\delta^n X_2)\cdot 1 \end{bmatrix}$$

In this case for the component which is clipped, the cutting operation must be done in a different way – the rounding function must be canceled in the equation **(4.79)**, and number one with the appropriate sign must be put instead.

Besides, Figures 4.8 and 4.9 prove that the range clipping, which may happen when the inverse transform is done, does not affects proposition 2 (refer to Section 4.4), which is presented in Section 4.4. Hence the maximum errors due to single encoding/decoding cycle cannot be greater then the upper bounds mentioned in that proposition.

The cases presented in Figures 4.8 and 4.9 may be considered special cases of the general cutting operation, defined in Equation **(4.81)** for the increment of the second component:

$$\begin{bmatrix} \delta^{n+1}\hat{X}_1 \\ \delta^{n+1}\hat{X}_2 \end{bmatrix} = -\begin{bmatrix} \delta^{n}X_1 \\ \delta^{n}X_2 \end{bmatrix} + \begin{bmatrix} \text{round}\left(\delta^{n}X_1\right) \\ \text{sign}(\delta^{n}X_2)\cdot C_2 \end{bmatrix}, \qquad \textbf{(4.82)}$$

where $C_k \in [0,1,...,Max_k]$ and $Max_k = \left\lfloor \delta X_k^{\max} \right\rfloor = \left\lfloor \frac{1}{2}\cdot\sum_{i=1}^{N}\left|s_{k,i}\right| \right\rfloor$.

Value C_k tells how far away from the edge of the range cuboid its data point is located. Hence the equation **(4.82)** shows, how to modify the equation **(4.79)** in order to take into account a range clipping for a certain component. No matters what $\delta^n X_k$ and corresponding C_k are like, every increment $\delta^{n+1}\hat{X}_k$ which has been cut, belongs either to the interval $[-\delta X_k^{\max}, 0]$ or to the interval $[0, \delta X_k^{\max})$. Such a situation for $k = 2$ is presented in Fig. 4.10.

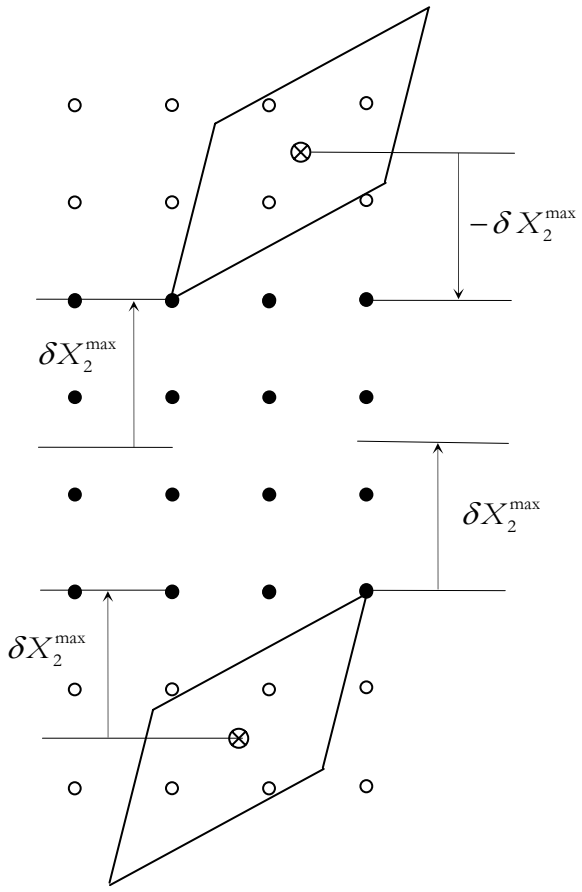

Fig 4.10 The greatest clipping error for maximum and minimum value of component X_2.

The parallelogram, drawn on Fig. 4.10, is the elementary cell of Y frame of reference for the two-dimensional transform. The parallelepipeds show the most critical locations of the elementary cell in Y frame of reference, as long as the X_2 component is considered. For both the cases presented on the figure C_k is equal to zero, but might also be equal to one, because $Max_2 = \lfloor \delta X_2^{\max} \rfloor = 1$.

4.6.3 Consecutive cycles analysis – approach 1

In Section 4.6.1 the system of two recursive equations **(4.68)** and **(4.79)** has been derived. Let us recall it:

$$\delta^{n+1}\vec{\hat{\mathbf{X}}} = -\delta^{n}\vec{\mathbf{X}} + \text{round}\,\textit{and / or}\,\text{cut}\left(\delta^{n}\vec{\mathbf{X}}\right), \text{ for n=0, 1, 2, …} \quad \textbf{(4.83)}$$

$$\delta^{n}\vec{\mathbf{X}} = -\delta^{n}\vec{\hat{\mathbf{X}}} + \mathbf{T}^{-1}\cdot\text{round}\left(\mathbf{T}\cdot\,\delta^{n}\vec{\hat{\mathbf{X}}}\right), \text{ for n=1, 2, …} \quad \textbf{(4.84)}$$

This system of difference equations makes it possible to trace the migration of the data point in the data space during multiple encoding/decoding cycles by means of increments, as shown in **(4.85)**.

$$\delta^{0}\vec{\mathbf{X}} \xrightarrow{\textbf{(4.83)}} \delta^{1}\vec{\hat{\mathbf{X}}} \xrightarrow{\textbf{(4.84)}} \delta^{1}\vec{\mathbf{X}} \xrightarrow{\textbf{(4.83)}} \delta^{2}\vec{\hat{\mathbf{X}}} \ldots \quad \textbf{(4.85)}$$

Hence one does not need to know the very components of the considered data point to learn how it is going to evolve. Few other values must be known instead:

1. $\delta^{0}\vec{\mathbf{X}}$ – the distance between the original data point (the elementary cell center in X frame of reference) and the nearest elementary cell center in Y frame of reference,
2. the distances between the original data point and the "nearest" boundaries of the X range cuboid.

The first item gives the starting point. The second one tells how to modify the equation **(4.83)** in order to include the range clipping effect in the reasoning. Migration continues till the secondary migration vector consists of zeroes only. During the first encoding/decoding cycle the operation of cutting must be done with respect to the value C_n, which is defined in **(4.82)**. During the subsequent cycles the cutting with $C_n = 0$ will only occur, because the clipping operation makes the data point migrate onto the hyper-edge of the range hyper-cuboid, and will never make the data point enter it. In other words, after the first cycle the primary migration vector will always be parallel to the considered hyper-edge of the range hyper-cuboid. The secondary migration vector can make the data point enter the range hyper-cuboid and then migration ends because $\mathrm{L}_{\infty}(\mathbf{T}) < 1$ (refer to proposition 3, Section 4.5). To decide which hyper-edges can be considered "nearest" it is necessary to compare the distance between the considered data point and hyper-edge with the maximum possible rounding error, which is found according to the

proposition 2 (refer to Section 4.4). Accumulated primary migration vectors for the consecutive cycles show how far away from its original value the data point has migrated and makes it possible to estimate whether another hyper-edge is reached before the migration along a certain hyper-edge ends.

To analyze the evolution of a data point for every possible case it is necessary to consider all the possible $\delta^0\vec{\mathbf{X}}$ vectors which may occur in the vicinity of every kind of the range hyper-cuboid cuboid hyper-edge. Now a two-dimensional case will be analyzed because it is easy to illustrate it using drawings. It is very easy to extend this reasoning to an N-dimensional case. Thus let us define a two-dimensional transform:

$$\begin{bmatrix} \hat{Y}_1^0 \\ \hat{Y}_2^0 \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} X_1^0 \\ X_2^0 \end{bmatrix}. \qquad \textbf{(4.86)}$$

Fig. 4.11 shows all the possible $\delta^0\vec{\mathbf{X}}$ vectors which may occur in the vicinity of a two-dimensional range cuboid edge with $X_2^0 = X_2^{\max}$ (thus $C_2 = 0$), for a certain two-dimensional linear transform. The parallelogram, which is drawn on Fig. 4.10, is the elementary cell of Y frame of reference for this two-dimensional transform, and implies the shape of the figure which contains all the endings of considered $\delta^0\vec{\mathbf{X}}$ vectors.

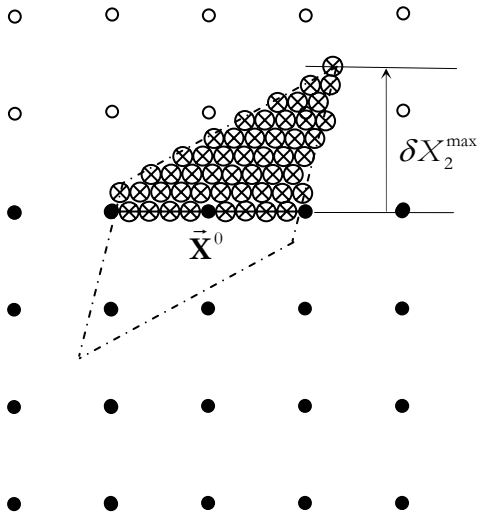

Fig 4.11. All possible $\delta^0\vec{\mathbf{X}}$ vectors, which stick out of the X range cuboid by not more then $\delta X_2^{\max}$, for $\vec{\mathbf{X}}^0$ belonging to the edge of X range cuboid

All possible $\delta^0\vec{\mathbf{X}}$ vectors start in the black dot denoted as $\vec{\mathbf{X}}^0$ and end in any of the empty crossed circles. Fig. 4.12 shows the same situation for various $\vec{\mathbf{X}}^0$. Fig. 4.12 shows all possible $\delta^0\vec{\mathbf{X}}$ vectors which may occur in the vicinity of a two-dimensional range cuboid edge with $X_2^0 = X_2^{\max} - 1$ (thus $C_2 = 1$), for the same two-dimensional linear transform.

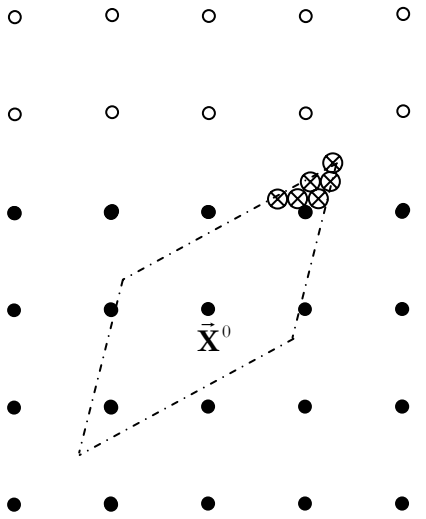

Fig 4.12. All possible $\delta^0\vec{\mathbf{X}}$ vectors, which stick out of the X range cuboid by not more then $\delta X_2^{\max}$, for $\vec{\mathbf{X}}^0$ belonging to the interior of range cuboid

The parallelograms, drawn on both the above figures, are confined by lines, which are described by the following equations:

$$\begin{aligned}
&\hat{Y}_1^0 + \frac{1}{2} = t_{11} \cdot \hat{X}_1^1 + t_{12} \cdot \hat{X}_2^1, \\
&\hat{Y}_1^0 - \frac{1}{2} = t_{11} \cdot \hat{X}_1^1 + t_{12} \cdot \hat{X}_2^1, \\
&\hat{Y}_2^0 + \frac{1}{2} = t_{21} \cdot \hat{X}_1^1 + t_{22} \cdot \hat{X}_2^1, \\
&\hat{Y}_2^0 - \frac{1}{2} = t_{21} \cdot \hat{X}_1^1 + t_{22} \cdot \hat{X}_2^1.
\end{aligned} \qquad (4.87)$$

The interior of such a parallelogram can be defined by the following system of inequalities:

$$\begin{cases}
\hat{Y}_1^0 + \frac{1}{2} > t_{11} \cdot \hat{X}_1^1 + t_{12} \cdot \hat{X}_2^1 \\
\hat{Y}_1^0 - \frac{1}{2} < t_{11} \cdot \hat{X}_1^1 + t_{12} \cdot \hat{X}_2^1 \\
\hat{Y}_2^0 + \frac{1}{2} > t_{21} \cdot \hat{X}_1^1 + t_{22} \cdot \hat{X}_2^1 \\
\hat{Y}_2^0 - \frac{1}{2} < t_{21} \cdot \hat{X}_1^1 + t_{22} \cdot \hat{X}_2^1
\end{cases} \qquad (4.88)$$

All the ends of the considered $\delta^0 \vec{\mathbf{X}}$ vectors are located in the area, which is defined by the system of inequalities, which defines the upper part of the parallelogram:

$$\begin{cases}
\hat{Y}_1^0 + \frac{1}{2} \geq t_{11} \cdot \hat{X}_1^1 + t_{12} \cdot \hat{X}_2^1, \\
X_2^{\max} \leq \hat{X}_2^1, \\
\hat{Y}_2^0 + \frac{1}{2} \geq t_{21} \cdot \hat{X}_1^1 + t_{22} \cdot \hat{X}_2^1, \\
\hat{Y}_2^0 - \frac{1}{2} < t_{21} \cdot \hat{X}_1^1 + t_{22} \cdot \hat{X}_2^1.
\end{cases} \qquad (4.89)$$

It is convenient to rearrange the above set of inequalities in a following way:

$$\begin{cases} \hat{Y}_1^0 + \dfrac{1}{2} \geq t_{11} \cdot \hat{X}_1^1 + t_{12} \cdot \hat{X}_2^1, \\ \hat{Y}_2^0 + \dfrac{1}{2} \geq t_{21} \cdot \hat{X}_1^1 + t_{22} \cdot \hat{X}_2^1, \\ X^{\max} \leq \hat{X}_2^1, \\ \hat{Y}_2^0 - \dfrac{1}{2} < t_{21} \cdot \hat{X}_1^1 + t_{22} \cdot \hat{X}_2^1. \end{cases} \tag{4.90}$$

The appropriate equation from **(4.86)** is subtracted from both sides of the first, second and fourth inequality.

$$\begin{cases} \hat{Y}_1^0 + \dfrac{1}{2} - \hat{Y}_1^0 \geq t_{11} \cdot (\hat{X}_1^1 - X_1^0) + t_{12} \cdot (\hat{X}_2^1 - X_2^0), \\ \hat{Y}_2^0 + \dfrac{1}{2} - \hat{Y}_2^0 \geq t_{21} \cdot (\hat{X}_1^1 - X_1^0) + t_{22} \cdot (\hat{X}_2^1 - X_2^0), \\ X^{\max} - X^{\max} \leq \hat{X}_2^1 - X^{\max}, \\ \hat{Y}_2^0 - \dfrac{1}{2} - \hat{Y}_2^0 < t_{21} \cdot (\hat{X}_1^1 - X_1^0) + t_{22} \cdot (\hat{X}_2^1 - X_2^0). \end{cases} \tag{4.91}$$

Hence the system of inequalities for the increments is obtained:

$$\begin{cases} \dfrac{1}{2} \geq t_{11} \cdot \delta^0 X_1 + t_{12} \cdot \delta^0 X_2, \\ \dfrac{1}{2} \geq t_{21} \cdot \delta^0 X_1 + t_{22} \cdot \delta^0 X_2, \\ 0 \leq \delta^0 X_2 - C_2, \\ -\dfrac{1}{2} < t_{21} \cdot \delta^0 X_1 + t_{22} \cdot \delta^0 X_2. \end{cases} \tag{4.92}$$

Thus we get:

$$\begin{cases} \dfrac{1}{2} \geq t_{11} \cdot \delta^0 X_1 + t_{12} \cdot \delta^0 X_2, \\ \dfrac{1}{2} \geq t_{21} \cdot \delta^0 X_1 + t_{22} \cdot \delta^0 X_2, \\ C_2 \leq \delta^0 X_2, \\ -\dfrac{1}{2} < t_{21} \cdot \delta^0 X_1 + t_{22} \cdot \delta^0 X_2. \end{cases} \tag{4.93}$$

Of course the above system of inequalities is not contradictory to the following one:

$$\begin{cases} \frac{1}{2} \geq t_{11} \cdot \delta^0 X_1 + t_{12} \cdot \delta^0 X_2 = \delta^0 \hat{Y}_1 , \\ \frac{1}{2} \geq t_{21} \cdot \delta^0 X_1 + t_{22} \cdot \delta^0 X_2 = \delta^0 \hat{Y}_2 , \\ -\frac{1}{2} < t_{11} \cdot \delta^0 X_1 + t_{12} \cdot \delta^0 X_2 = \delta^0 \hat{Y}_1 , \\ -\frac{1}{2} < t_{21} \cdot \delta^0 X_1 + t_{22} \cdot \delta^0 X_2 = \delta^0 \hat{Y}_2 , \\ C_2 \leq \delta^0 X_2 . \end{cases} \quad \textbf{(4.94)}$$

The above system of inequalities can be rewritten as follows:

$$\begin{cases} \frac{1}{2} \geq \delta^0 \hat{Y}_1 , \\ \frac{1}{2} \geq \delta^0 \hat{Y}_2 , \\ -\frac{1}{2} < \delta^0 \hat{Y}_1 , \\ -\frac{1}{2} < \delta^0 \hat{Y}_2 , \\ C_2 \leq \delta^0 X_2 . \end{cases} \quad \textbf{(4.95)}$$

And hence we have:

$$\begin{cases} \delta^0 \hat{Y}_1 \in (-0.5, 0.5], \\ \delta^0 \hat{Y}_2 \in (-0.5, 0.5], \\ C_2 \leq \delta^0 X_2 . \end{cases} \quad \textbf{(4.96)}$$

In order to consider all the $\delta^0 \vec{\mathbf{X}}$ vectors which may occur when clipping the X_2 component, it is necessary to take into account the parallelograms defined by **(4.96)** for all the possible values of C_2. If another components are to be clipped then the appropriate inequalities for these components will have to be included into **(4.96)**.

For N dimensions the clipping errors may happen when the integer **Y** coordinates stick out of the **X** range cuboid by the distances which belong to the following intervals (described in the new frame of reference **Y**):

$$\begin{cases} \delta^0 \hat{Y}_1 \in (-0.5, 0.5], \\ \delta^0 \hat{Y}_2 \in (-0.5, 0.5], \\ \ldots, \\ \delta^0 \hat{Y}_k \in (-0.5, 0.5], \\ \ldots, \\ \delta^0 \hat{Y}_N \in (-0.5, 0.5]. \end{cases} \tag{4.97}$$

The intervals defined above are implied by the first definition of rounding **(4.01)** ($\text{round}_1(x) = \lfloor x + 0.5 \rfloor$). To consider also the biggest possible clipping errors of **X** components (**Y** components are not clipped at all) it is necessary to take into account both the ends of the above intervals. As only one end of every interval belongs to it, it will be useful to define the value "*AlmostHalf*":

$$\begin{aligned} &AlmostHalf = 0.5 - \varepsilon, \\ &\text{where} \quad \varepsilon \to 0. \end{aligned} \tag{4.98}$$

When using floating point representation of real numbers *AlmostHalf* is equal to the biggest number smaller then half not rounded to half.

Instead of the intervals defined above we can use the intervals defined below for numeric calculations.

$$\begin{cases} \delta^0 \hat{Y}_1 \in [-1 \cdot AlmostHalf, Half], \\ \delta^0 \hat{Y}_2 \in [-1 \cdot AlmostHalf, Half], \\ \ldots, \\ \delta^0 \hat{Y}_k \in [-1 \cdot AlmostHalf, Half], \\ \ldots, \\ \delta^0 \hat{Y}_N \in [-1 \cdot AlmostHalf, Half]. \end{cases} \tag{4.99}$$

The above set of conditions must be supplied with the appropriate conditions for the clipped components:

$$\begin{aligned} &C_k \le \delta^0 X_k \ \ for \ X_k^0 = X_k^{\max} - C_k, \\ &-C_k \ge \delta^0 X_k \ \ for \ X_k^0 = X_k^{\min} + C_k. \end{aligned} \tag{4.100}$$

For every value C_k, for every clipped component the figure defined by the complete set of conditions must be sampled to get the endings of all the considered $\delta^0\vec{\mathbf{X}}$ vectors. This kind of reasoning is substantially excessive. A better solution will be presented later.

The resulting $\delta^0\vec{\mathbf{X}}$ vectors must substituted into the equation **(4.79)**. Let us recall it:

$$\begin{bmatrix} \delta^1\hat{X}_1 \\ \delta^1\hat{X}_2 \\ \dots \\ \delta^1\hat{X}_k \\ \dots \\ \delta^1\hat{X}_N \end{bmatrix} = -\begin{bmatrix} \delta^0 X_1 \\ \delta^0 X_2 \\ \dots \\ \delta^0 X_k \\ \dots \\ \delta^0 X_N \end{bmatrix} + \begin{bmatrix} \text{round}\left(\delta^0 X_1\right) \\ \text{round}\left(\delta^0 X_2\right) \\ \dots \\ \text{round}\left(\delta^0 X_k\right) \\ \dots \\ \text{round}\left(\delta^0 X_N\right) \end{bmatrix}. \qquad \textbf{(4.101)}$$

Hence in equation **(4.101)** for any component, which is clipped, one has to cancel corresponding rounding function and put the expression $\text{sign}(\delta^0 X_i)\cdot C_i$ instead. When considering clipping over the three-dimensional range cuboid face, the rounding must be canceled for one component, when considering three-dimensional range cuboid edge – for two components, and when considering the three-dimensional range cuboid corner – for three components. The choice of the components, for which this operation is to be done, determines which face, edge or corner of **X** range cuboid is to be considered. N-dimensional range hyper-cuboids where N is greater then three, have more types of confining figures then just a plane, line and point.

The operation of range clipping will never make the data point to enter interior of the X range cuboid. Entering the X range cuboid interior will be possible only after rounding in Y frame of reference. Hence, if range clipping occurs in the first encoding/decoding cycle then the data point will reach the hyper-edge of the X range cuboid. Then the data point migrates, as the primary migration vector shows, along the edge of the range cuboid. If such a migration were prolonged for many encoding/decoding cycles, the data point might reach the hyper-edge with dimensionality reduced bye one (corner of the range rectangle in this case). Let us temporarily disregard such an opportunity.

Let us rewrite the equation **(4.68)** for the first cycle:

$$\begin{bmatrix} \delta^1 X_1 \\ \delta^1 X_2 \\ \dots \\ \delta^1 X_k \\ \dots \\ \delta^1 X_N \end{bmatrix} = -\begin{bmatrix} \delta^1 \hat{X}_1 \\ \delta^1 \hat{X}_2 \\ \dots \\ \delta^1 \hat{X}_k \\ \dots \\ \delta^1 \hat{X}_N \end{bmatrix} + \mathbf{T}^{-1} \cdot \text{round}\left(\mathbf{T} \cdot \begin{bmatrix} \delta^1 \hat{X}_1 \\ \delta^1 \hat{X}_2 \\ \dots \\ \delta^1 \hat{X}_k \\ \dots \\ \delta^1 \hat{X}_N \end{bmatrix} \right). \qquad \textbf{(4.102)}$$

The vectors $\delta^1 \hat{\mathbf{X}}$, which have been obtained according to **(4.101)**, are substituted into the equation **(4.102)**. As variables have sufficient precision for **Y** components range, there is no range clipping in this transformation, thus equation **(4.101)** is always valid. The resulting $\delta^1 \mathbf{X}$ vectors are substituted into the equation **(4.79)** for the second cycle:

$$\begin{bmatrix} \delta^2 \hat{X}_1 \\ \delta^2 \hat{X}_2 \\ \dots \\ \delta^2 \hat{X}_k \\ \dots \\ \delta^2 \hat{X}_N \end{bmatrix} = -\begin{bmatrix} \delta^1 X_1 \\ \delta^1 X_2 \\ \dots \\ \delta^1 X_k \\ \dots \\ \delta^1 X_N \end{bmatrix} + \begin{bmatrix} \text{round}\left(\delta^1 X_1\right) \\ \text{round}\left(\delta^1 X_2\right) \\ \dots \\ \text{round}\left(\delta^1 X_k\right) \\ \dots \\ \text{round}\left(\delta^1 X_N\right) \end{bmatrix}. \qquad \textbf{(4.103)}$$

Here again both rounding and range clipping may occur. But a range clipping must be considered in a different way than before. As the data point has reached the boundary of the clipping cuboid in the first cycle, the value C_n is zero. Besides the data point might either approach or not a hyper-edge with reduced dimensionality in the previous cycle. If not, then those components which were rounded in a previous cycle are to be rounded in this cycle too. If yes, then some of the components which were rounded in the previous cycle will be cut in this cycle and the accumulated primary migration vectors tell which ones. Those components, which were clipped in a previous cycle, may be either clipped or rounded in this cycle. It depends both on the signs and the magnitudes of the corresponding errors ($\delta^0 X_1, \delta^1 X_1, \delta^0 X_2, \delta^1 X_2, \dots, \delta^0 X_N, \delta^1 X_N$).

Let us consider an exemplary case. Let us assume that component X_1 was clipped in the first cycle (equation **(4.101)**). Thus if $\delta^0 X_1$ and $\delta^1 X_1$ have the same signs, and absolute value of $\delta^1 X_1$ is not smaller than half, then range clipping will occur. In all the other cases rounding will occur. Resulting $\delta^2 X_1, \delta^2 X_2, ..., \delta^2 X_N$ errors are substituted into the equation **(4.57)** for the third cycle. Consecutive cycles are following the same pattern. The error accumulation for the explored case ends when the *n*-th secondary migration vector:

$$\text{round}\left(\mathbf{T}\cdot\begin{bmatrix}\delta^n\hat{X}_1\\ \delta^n\hat{X}_2\\ \cdots\\ \delta^n\hat{X}_k\\ \cdots\\ \delta^n\hat{X}_N\end{bmatrix}\right)=\vec{0}. \qquad \textbf{(4.104)}$$

4.6.4 Consecutive cycles analysis – approach 2

As the reasoning, which is presented in Section 4.6.2, is substantially excessive it is desired to find a better one. Let us choose the $\delta^1\vec{\hat{\mathbf{X}}}$ as a starting point. The two-dimensional case is now considered, which can be easily extended to more dimensions. One can estimate all the different $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which stick out of the *X* range cuboid in positive X_2 direction, and choose those critical ones for further calculations. For the negative X_2 direction only the increment sign must be changed. In order to search for such $\delta^1\vec{\hat{\mathbf{X}}}$ vectors let us find all the possible $\delta^0\vec{\mathbf{X}}$. This is presented in Fig. 4.13 which is recalled below.

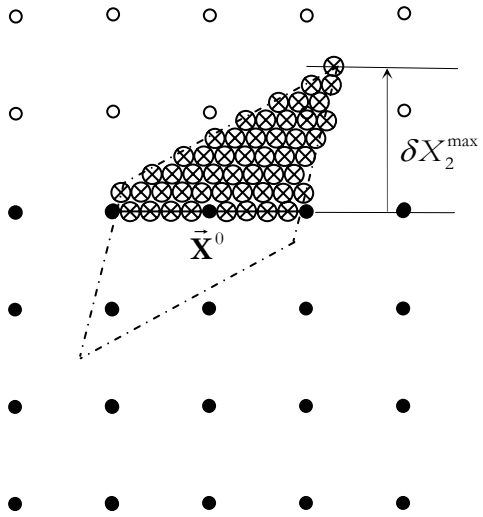

Fig 4.13. All possible $\delta^0\vec{\mathbf{X}}$ vectors, which stick out of the X range cuboid by not more then $\delta X_2^{\max}$, for $\vec{\mathbf{X}}^0$ belonging to the edge of range cuboid.

The $\delta^0\vec{\mathbf{X}}$ vectors start in the black dot denoted as $\vec{\mathbf{X}}^0$ and end in any of the empty crossed circles. Fig. 4.14 shows the same situation for various $\vec{\mathbf{X}}^0$.

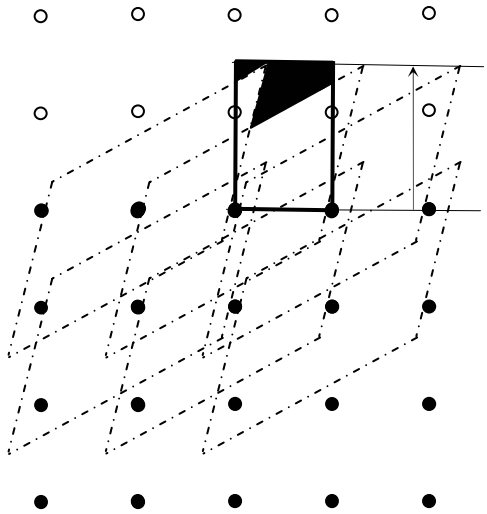

Fig 4.14. All possible $\delta^1\vec{\hat{\mathbf{X}}}$ *vectors, which stick out of the X range cuboid by not more then* $\delta X_2^{\max}$.

It can be easily seen that the rectangle, drawn in Fig. 4.14, with the shaded region excluded, contains all the possible different locations for the empty crossed circles. Hence any of the different $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which stick out of the *X* range cuboid in positive X_2 direction, starts in any empty crossed circle belonging to the above defined region, and ends in the nearest black dot. The rectangle drawn in Fig. 4.14 is a two-dimensional case of a **clipping hyper-cuboid**.

Definition: the *n*-th clipping hyper-cuboid is the smallest hyper-cuboid which contains all the possible $\delta^n\vec{\mathbf{X}}$ vectors.

Hence it is more than enough to consider the $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which start in the clipping cuboid and end in the nearest black dot. Thus one can see, this reasoning is still excessive, but not as much as the former one. This clipping cuboid is shown in Fig. 4.15.

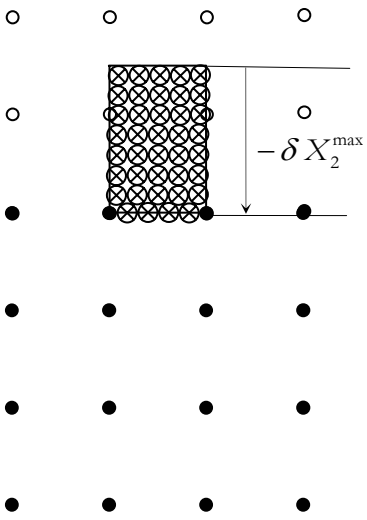

Fig 4.15. All possible $\delta^{n+1}\vec{\hat{\mathbf{X}}}$ *vectors, which stick out of the X range cuboid by not more then* $\delta X_2^{\max}$

The clipping cuboid, which is shown on Fig. 4.15, contains all the $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which should be taken into account. Once having these vectors one can use the recursive formulas **(4.83)** and **(4.84)** as it was discussed before. However it is not necessary to consider all these vectors. One can chose only the critical ones for further analysis. In order to find the critical cases let us redraw the fragment of clipping cuboid with the height $h \in [0, \delta X_2^{\max}]$ and consider only those crossed circles, which are lying on its upper edge. Note the interval defining h is closed in this case.

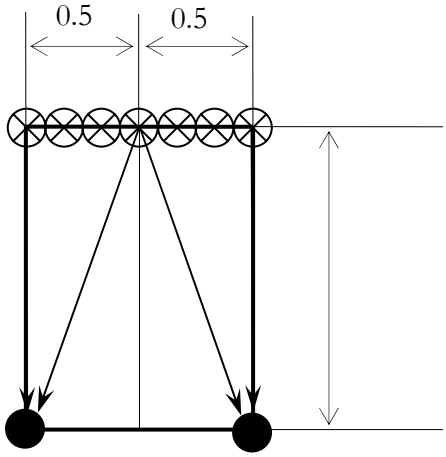

Fig. 4.16 The fragment of two-dimensional clipping cuboid.

Fig. 4.16 shows all the considered vectors for given h. As h ranges from 0 to $\delta X_2^{\max}$, all the considered vectors are taken into account. The same result can be achieved in another way, as shown on Fig. 4.17.

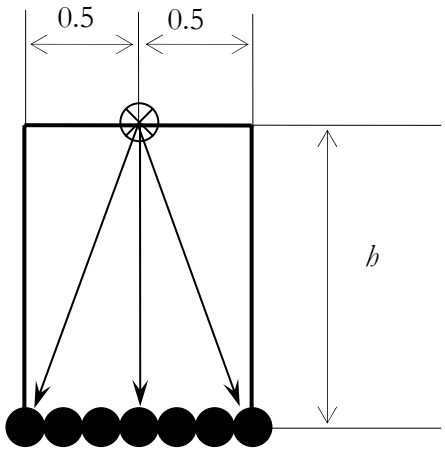

Fig. 4.17 The fragment of two-dimensional clipping cuboid.

Fig. 4.17 contains exactly the same set of vectors as Fig. 4.16. For the reasons, which are discussed later, in the cases when $h \in [0, \delta X_2^{\max})$ it is enough to consider the two outer vectors, what is shown in Fig. 4.18. Note the interval stating the h range is half opened here. For $h = \delta X_2^{\max}$ all the vectors must be considered, as it is shown in Fig. 4.17.

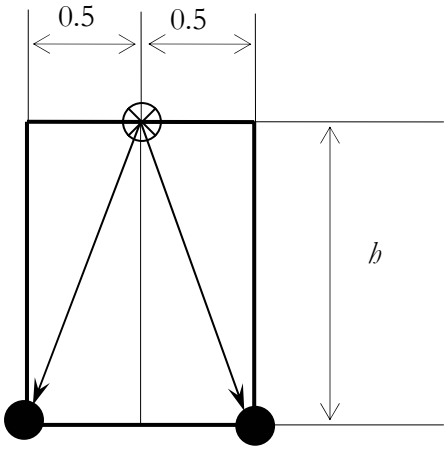

Fig. 4.18 The fragment of two-dimensional clipping cuboid.

Fig. 4.19 shows the whole clipping cuboid with those considered borders drawn with thick lines.

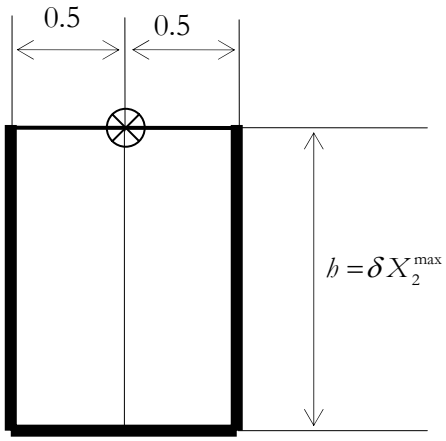

Fig. 4.19 The two-dimensional clipping cuboid.

The empty crossed circle on Fig. 4.19, in which the considered $\delta^1\vec{\hat{\mathbf{X}}}$ vectors start, is the first cycle Y elementary cell center, and the clipping cuboid is the first cycle clipping cuboid. It is shown later that construction of the *n*'th clipping cuboid is equivalent to estimation of the critical cases for the equation **(4.83)**, which is the second one of the two recursive equations. As stated above it is sufficient to consider only those $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which start at the crossed circle and end at any point belonging to that part of the clipping cuboid border, which is drawn with a thick line on Fig. 4.19. This is because the next step is to find all the resulting secondary migration vectors in either X or Y frame of reference according to the definitions **(4.69)** and **(4.70)**, which are recalled below with $n = 1$:

$$\delta^1\vec{\hat{\mathbf{X}}}^{Mig2} = \mathbf{T}^{-1} \cdot \mathrm{round}\left(\mathbf{T} \cdot \delta^1\vec{\hat{\mathbf{X}}}\right), \tag{4.105}$$

$$\delta^1\vec{\mathbf{Y}}^{Mig2} = \mathrm{round}\left(\mathbf{T} \cdot \delta^1\vec{\hat{\mathbf{X}}}\right). \tag{4.106}$$

The secondary migration vector is a constituent of the equation **(4.82)**, which is the first one of the two recursive equations. One does not need to use the very equation **(4.82)**, because the exact values of $\delta^n\vec{\mathbf{X}}$ vectors are not needed as they are estimated when finding the candidates for *n*'th elementary cell center. Fig. 4.20 illustrates the equation **(4.82)**.

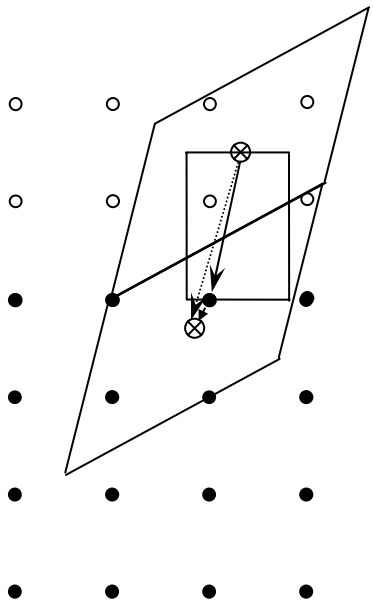

Fig 4.20. Two dimensional case - clipping rectangle.

Vectors $\delta^1\vec{\hat{\mathbf{X}}}$ start in an empty crossed circle and end in a black dot. Vector $\delta^1\vec{\mathbf{X}}$ starts in the black dot and ends in the empty crossed circle. The secondary migration vector starts and ends in the empty crossed circle. Hence it is clear that secondary migration vectors shows which Y elementary cell the data point will migrate to, due to range clipping. It depends in which Y elementary cell vector $\delta^1\vec{\hat{\mathbf{X}}}$ ends in. If any Y elementary cell were small enough to be contained inside the clipping rectangle, we would have to consider all the $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which start in the first cycle Y elementary cell center and end in the clipping cuboid. As it is not the case it is enough to consider only the boundaries of the clipping cuboid. Since only the transforms $\mathbf{T}$ satisfying the condition $\mathrm{L}_\infty(\mathbf{T}) < 1$ are considered, it is not necessary to consider the upper boundary of the clipping cuboid. This is why, when looking for all the secondary migrations vectors, it is necessary to check only those $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which start at the crossed circle and end at any point belonging to that part of the clipping cuboid border, which is drawn with a thick line

on Fig. 4.19. All the resulting migrations vectors must be subsequently added to the first cycle Y elementary cell center, giving the various second cycle Y elementary cell centers. In this moment the analysis of the second encoding decoding cycle starts. For every such a candidate the appropriate clipping cuboid must be constructed, another set of migrations vectors found and so on until the secondary migration vector consists of zeroes. Let us construct the clipping cuboid for N-dimensional X range hyper-cuboid over it hyper-edge with dimensionality (N-1) and with maximum value of its X_n coordinate. As shown in **(4.97)**, the clipping errors may happen when the integer **Y** coordinates stick out of the **X** range cuboid by the distances which belong to the following intervals (described in the new frame of reference **Y**):

$$\begin{cases} \delta^0 \hat{Y}_1 \in (-0.5, 0.5], \\ \delta^0 \hat{Y}_2 \in (-0.5, 0.5], \\ \ldots, \\ \delta^0 \hat{Y}_k \in (-0.5, 0.5], \\ \ldots, \\ \delta^0 \hat{Y}_N \in (-0.5, 0.5]. \end{cases} \tag{4.107}$$

For numeric calculations the intervals defined **(4.97)** may be used instead:

$$\begin{cases} \delta^0 \hat{Y}_1 \in [-1 \cdot AlmostHalf, 0.5], \\ \delta^0 \hat{Y}_2 \in [-1 \cdot AlmostHalf, 0.5], \\ \ldots, \\ \delta^0 \hat{Y}_k \in [-1 \cdot AlmostHalf, 0.5], \\ \ldots, \\ \delta^0 \hat{Y}_N \in [-1 \cdot AlmostHalf, 0.5]. \end{cases} \tag{4.108}$$

The first stage is to consider a set of an N – dimensional vectors. Each vector in this set has either *Half* or *–AlmosHalf* in every component in **Y** coordinate system. Then all these vectors must be transformed to **X** coordinate system according to **(4.18)**:

$$\begin{bmatrix} \delta^0 X_1 \\ \delta^0 X_2 \\ \dots \\ \delta^0 X_k \\ \dots \\ \delta^0 X_N \end{bmatrix} = \mathbf{T}^{-1} \cdot \begin{bmatrix} \delta^0 \hat{Y}_1 \\ \delta^0 \hat{Y}_2 \\ \dots \\ \delta^0 \hat{Y}_k \\ \dots \\ \delta^0 \hat{Y}_N \end{bmatrix}. \tag{4.109}$$

Let us assume that the (N-1) – dimensional hyper-edge with maximum k-th component is considered. The vector $\delta^0\mathbf{X}$ which has the greatest positive value in its k-th increment $\delta^0 X_k = \delta X_k^{\max} = \frac{1}{2} \cdot \sum_{i=1}^{N} \left| s_{k,i} \right|$ must be chosen. Hence, the most critical case for the first cycle Y elementary cell center is found. The clipping cuboid vertices for the first cycle Y elementary cell have the following coordinates:

$$\begin{bmatrix} \delta^0 X_1 - 0.5 \; or \; \delta^0 X_1 + 0.5 \\ \delta^0 X_2 - 0.5 \; or \; \delta^0 X_2 + 0.5 \\ \dots \\ 0 \; or \; \delta \; X_k^{\max} \\ \dots \\ \delta^0 X_N - 0.5 \; or \; \delta^0 X_N + 0.5 \end{bmatrix}, \tag{4.110}$$

what results in the following values of critical $\delta^1\hat{\mathbf{X}}$ vectors components:

$$\begin{bmatrix} -0.5 \; or \; 0.5 \\ -0.5 \; or \; 0.5 \\ \dots \\ -\delta \; X_k^{\max} \; or \; 0 \\ \dots \\ -0.5 \; or \; 0.5 \end{bmatrix}. \tag{4.111}$$

Hence the most critical first cycle Y elementary cell clipping cuboid for the $X_k^0 = X_k^{\max}$ hyper-edge has been constructed. The borders of the clipping cuboid must be densely sampled to get all the possible secondary migration vectors. In this way not only the most critical location of first cycle Y elementary cell center ($\delta^0 X_k = \delta X_k^{\max}$) is considered, but also all the other possible ones. This is because $\delta^1 \hat{X}_k$ varies between $-\delta \; X_k^{\max}$ and zero. One can easily see that, the presented

in (4.111), upper bounds for values of $\delta^1\hat{\mathbf{X}}$ vector components are the same as those, which can be deduced from the equation (4.84), which is recalled below:

$$\begin{bmatrix} \delta^1\hat{X}_1 \\ \delta^1\hat{X}_2 \\ \dots \\ \delta^1\hat{X}_k \\ \dots \\ \delta^1\hat{X}_N \end{bmatrix} = -\begin{bmatrix} \delta^0 X_1 \\ \delta^0 X_2 \\ \dots \\ \delta^0 X_k \\ \dots \\ \delta^0 X_N \end{bmatrix} + \begin{bmatrix} \text{round}\left(\delta^0 X_1\right) \\ \text{round}\left(\delta^0 X_2\right) \\ \dots \\ 0 \\ \dots \\ \text{round}\left(\delta^0 X_N\right) \end{bmatrix}. \qquad (4.112)$$

The cutting of the k-th component with $C_k = 0$ is applied in equation (4.112). The further reasoning is the same as the one which was presented before.

4.6.5 Consecutive cycles analysis – approach 3

So far it has been shown that it is necessary to consider all the $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which start at the crossed circle and end at any point belonging to that part of the clipping cuboid border, which is drawn with a thick line on Fig. 4.19. However, this is still much. It is possible to reduce the number of considered $\delta^1\vec{\hat{\mathbf{X}}}$ vectors even more. If the considered clipping cuboid were contained in the convex hyper-solid, then it would be enough to consider only its vertices, hence it would be enough to consider only those $\delta^1\vec{\hat{\mathbf{X}}}$ vectors, which start in the considered elementary cell center, and end in the mentioned above clipping cuboid vertices. These vertices of the clipping cuboid are presented of Fig. 4.21

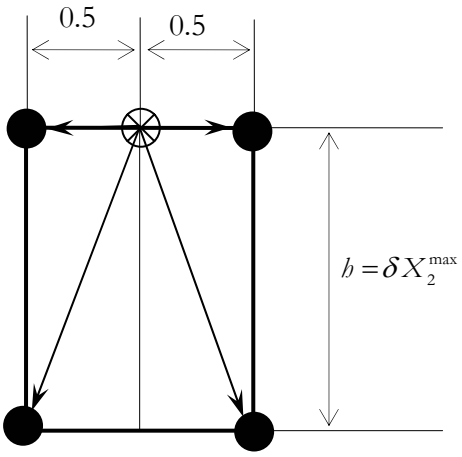

Fig. 4.21 The two-dimensional clipping cuboid.

As $L_\infty(\mathbf{T}) < 1$ then the upper vertices of the clipping cuboid do not need to be considered. The Y cell, in the center of which the $\delta^1\vec{\hat{\mathbf{X}}}$ vectors start and those, in which the clipping cuboid vertices are located, form a hyper-solid. If it is convex already, then no other secondary migration vectors will occur. If it is not, then that hyper-solid will have to be supplied with the minimum number of Y cells, which will make it convex. For further considerations those additional Y elementary cells and the migration vectors which end in their centers must be taken into account.

As the secondary migration vectors provide all the information about the migration due to range clipping it is enough to consider them all, without taking into account all possible $\delta^0\vec{\mathbf{X}}$ or $\delta^1\vec{\hat{\mathbf{X}}}$ vectors. The number of these migration vectors depends only on the location of the first cycle Y elementary cell relative to the hyper-edges of X range cuboid. Hence for every hyper-edge one can construct a directed graph which has a tree-like structure, provided each secondary migration vector has a different direction (the data point cannot return to the previous Y elementary cell). Such a directed graph may have cycles but may not have circuits. Y elementary cell centers are nodes in such a graph, and secondary migration vectors are its arcs. The first encoding/decoding cycle Y cell center is the root of such a quasi-tree. Note it is crucial to distinguish the encoding/decoding **cycle** from the **cycle** in a graph. The context makes it clear which meaning is used. The secondary migration vectors, being arcs, connect the root with the

closest nodes, being all the possible second cycle Y cell centers. Other arcs connect these nodes with other nodes keeping the graph structure. The distances between the first encoding/decoding cycle Y cell center and the closest hyper-edges, determine the number of accessible arcs and decides, which arc is to be chosen. Such a quasi-tree can be called "a directed rooted tree with cycles". This term needs some comments because the graph theory has not a commonly recognized nomenclature. Some authors define a tree as "a graph without cycles", the others however, use the term "tree with cycles", which is useful in the case described.

Let us assume that the first encoding/decoding cycle Y elementary cell is located in the vicinity of the N-1 dimensional hyper-edge of the X range hyper-cuboid and far away from the other hyper-edges. This determines the number of all the possible secondary migration vectors. Then the $\delta^1\vec{\hat{\mathbf{X}}}$ vector between the first encoding/decoding cycle elementary Y cell center and the closest X cell center decides which arc is to be chosen. However if we consider all the arcs (all the secondary migration vectors) possible in this case, we do not need to know this $\delta^1\vec{\hat{\mathbf{X}}}$ vector. If the data point reaches the node (Y elementary cell center) which is distant from the X range cuboid by the distance not greater then 0.5 in X frame of reference, then the error accumulation will end. The longest path between the critical location of the first encoding/decoding cycle Y elementary cell center and Y elementary cell center in which the data point migration stops gives the maximum number of encoding/decoding cycles which may be necessary to achieve the error saturation for a considered hyper-edge.

The exemplary case of such a graph for a two dimensional hyper-edge of the three dimensional X range cuboid is presented on Fig. 4.22. Each node in this graph is a three-dimensional Y cell center, which is denoted by increments in Y frame of reference. Increments (0.5, 0.5, 0.5) are the greatest possible ones but any others, with their absolute value not grater then 0.5, can also be substituted. Each arc in this graph represents a secondary migration vector

in Y frame of reference. In this example it is assumed that three graph arcs start in each node: (0 1 0), (0 1 –1) and (0 0 –1).

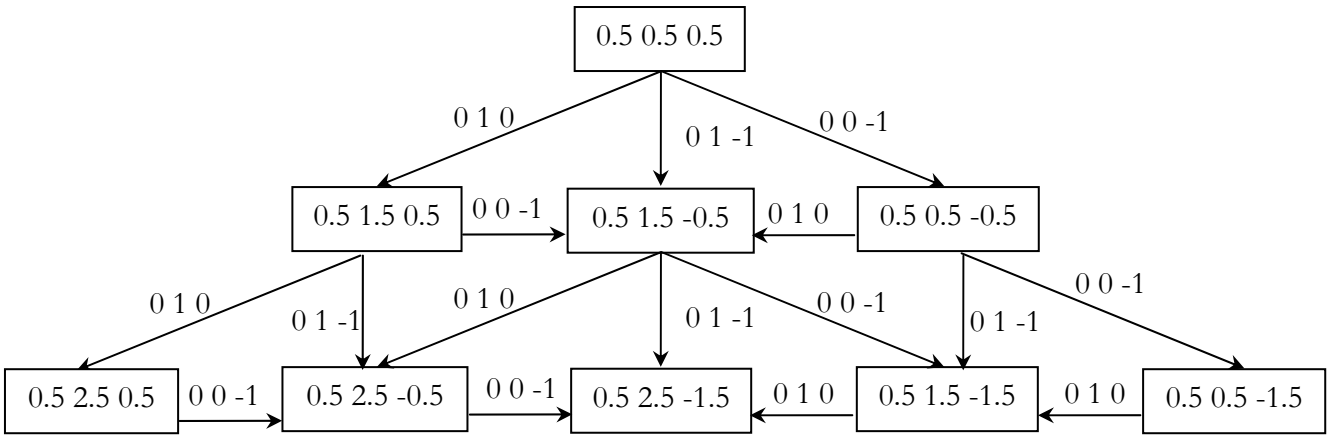

Fig. 4.22 Graph describing error accumulation, data points coordinates in nodes, secondary migration vectors in arcs.

For a given first (top) node and for a given two dimensional hyper-edge of the three dimensional X range cuboid, the sequence of nodes, which are visited during the data point migration, is a solution of the system of difference equations **(4.83)** and **(4.84)**, after transforming each node to the X frame of reference. For each different top node a different path in the above graph will be chosen. The longest path in this graph gives the maximum number of encoding/decoding cycles which may be necessary to achieve the error saturation for a given side face of the X range cuboid.

The graphs for different X range cube faces can be connected with each other creating much more complex graph, possibly having circuits. The problem of these interconnections is not resolved in this thesis.

The ideas introduced in this section are used in Chapter V, in which the color transform is discussed.

V. COLOR TRANSFORMATION

5.1 Properties of 8-bit YC_BC_R color transformation

Linear color transformations are important examples of linear transformations which are used in image processing. They usually transform *RGB* color representation, which is used in image acquisition, into another one, for example into YC_BC_R or opponent color representation. Matrices describing such transformations are square matrices of order three. As integer numbers are often used for such calculations, the transformations of this kind cause some errors due to rounding. Such a rounding may be considered as a vector quantization in the *RGB* color space or scalar quantization in a new one.

It is proved that the error accumulation for 8-bit representation of all the components for both *RGB* and YC_BC_R color spaces is limited to the first encoding/decoding cycle in most cases (see table 5.1), and even if some further errors originate in the next cycle the number of pixels changed then will be extremely small. The experiments were done for typical natural images with 8-bit red, green and blue color components. Results for three of them are presented in Table 5.1. The term "*All RGB*" in this table means that the image containing all possible RGB triples for 8-bit representation has been checked (the dynamic range was 255). It proves that no additional rounding errors are introduced into the image after the second encoding/decoding cycle for 8-bit representation of the color components.

TABLE 5.1 PERCENTAGE OF UNCHANGED IMAGE PELS – R, G AND B DYNAMIC RANGE (0, 255).

Image	Cycle number	Red	Green	Blue
Airplane	*1*	*58,638*	*73,337*	*49,311*
	2	*100*	*100*	*100*
Baboon	*1*	*58,407*	*73,484*	*49,529*
	2	*100*	*99,999*	*100*
	3	*100*	*100*	*100*
Tiffany	*1*	*67,628*	*74,927*	*49,898*
	2	*99,392*	*99,515*	*99,994*
	3	*100*	*100*	*100*
All RGB	*1*	*58,473*	*73,526*	*49,540*
	2	*99,991*	*99,987*	*99,964*
	3	*100*	*100*	*100*

More general analysis of error accumulation in YC_BC_R color transformation, which does not depend on an input data dynamic range, is presented in the following sections.

5.2 Error accumulation for YC_BC_R color transformation in the absence of a range clipping

Color transformation may be considered as **change of the frame of reference** in the *RGB* color space. As coordinates in both frames are integer, the lattice of discrete points originates – one for each frame. Every frame of reference has its own **base vectors** and its own **elementary cells** built on those vectors and shifted in such a way that the lattice points are located in their centers.

All possible values of input data are included in a range cuboid of the *RGB* color space due to the limited dynamic range of the original color components. Red, green and blue components have the same dynamic range thus the range cuboid turns into a range cube. There are many such transforms and YC_BC_R transformation is probably the one which is most commonly used. Forward transformation is defined by the equation **(5.01)**.

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \frac{1}{256} \begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \textbf{(5.01)}$$

Hence the matrix **T** is defined as follows:

$$\mathbf{T} = \frac{1}{256}\begin{bmatrix} 65.738 & 129.057 & 25.064 \\ -37.945 & -74.494 & 112.439 \\ 112.439 & -94.154 & -18.285 \end{bmatrix}, \qquad \textbf{(5.02)}$$

The inverse transformation is defined by the equation **(5.03)**:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{256}\begin{bmatrix} 298.082 & 0 & 408.583 \\ 298.082 & -100.291 & -208.120 \\ 298.082 & 516.411 & 0 \end{bmatrix}\begin{bmatrix} Y-16 \\ C_B-128 \\ C_R-128 \end{bmatrix} \qquad \textbf{(5.03)}$$

Hence the matrix **S** is defined as follows:

$$\mathbf{T}^{-1} \approx \mathbf{S} = \frac{1}{256}\begin{bmatrix} 298.082 & 0 & 408.583 \\ 298.082 & -100.291 & -208.120 \\ 298.082 & 516.411 & 0 \end{bmatrix} \qquad \textbf{(5.04)}$$

The **S** matrix is only an approximation of the $\mathbf{T}^{-1}$ matrix because of matrix coefficients rounding. To find the maximum dynamic range of the input data, for which this approximation is sufficient the product $(DynamicRange \cdot \mathbf{T} \cdot \mathbf{S})$ is to be studied.

$$1463 \cdot \mathbf{T} \cdot \mathbf{S} \approx \begin{bmatrix} 1463.00023 & 0.00156 & 0.00193 \\ -0.00000 & 1462.99585 & 0.00021 \\ -0.00000 & 0.00499 & 1462.99988 \end{bmatrix} \qquad \textbf{(5.05)}$$

The above result is shown with the accuracy of five decimal digits. It proves that if the accuracy of the two decimal digits for the red, green and blue component is used in the reasoning, then the drawn conclusions will be reliable as long as the dynamic range of the input integer data will not exceed 1463. It can be easily shown that for the greater dynamic range a better approximation of the $\mathbf{T}^{-1}$ matrix has to be used.

It is easy to find out that YC_BC_R color transformation satisfies neither the necessary condition (refer to Section 4.3) nor, in particular, the sufficient condition for the transformation reversibility (refer to Section 4.4). The sufficient condition for reversibility $\mathrm{L}_\infty(\mathbf{T}^{-1}) < 1$ is not satisfied, because $\mathrm{L}_\infty(\mathbf{S}) \approx 3.18 > 1$.

The necessary condition for reversibility $|\det(\mathbf{T})| \geq 1$ is not satisfied because $|\det(\mathbf{T})| \approx 0.16 < 1$.

Using the following equations (proposition 2, chapter 4.4):

$$\begin{aligned} Bnd_1 &= \text{round}[(|s_{11}|+|s_{12}|+...+|s_{1N}|)/2] \\ Bnd_2 &= \text{round}[(|s_{21}|+|s_{22}|+...+|s_{2N}|)/2] \\ &\cdots \\ Bnd_N &= \text{round}[(|s_{N1}|+|s_{N2}|+...+|s_{NN}|)/2] \end{aligned} \qquad \textbf{(5.06)}$$

it is possible to estimate the maximum rounding errors:

$$\begin{aligned} Bnd_R &= \text{round}\left[\frac{1}{2}\cdot\left(\frac{298.082}{256}+\frac{0}{256}+\frac{408.583}{256}\right)\right] = \text{round}[1.38...] = 1 \\ Bnd_G &= \text{round}\left[\frac{1}{2}\cdot\left(\frac{298.082}{256}+\frac{100.291}{256}+\frac{208.120}{256}\right)\right] = \text{round}[1.18...] = 1 \\ Bnd_B &= \text{round}\left[\frac{1}{2}\cdot\left(\frac{298.082}{256}+\frac{516.411}{256}+\frac{0}{256}\right)\right] = \text{round}[1.59...] = 2 \end{aligned} \qquad \textbf{(5.07)}$$

Hence during one encoding/decoding cycle the very rounding will introduce into the image the errors, which will not exceed one for the red and green components, and will not exceed two for the blue component.

The YC_BC_R color transformation satisfies the condition $\mathrm{L}_\infty(\mathbf{T}) < 1$, because $\mathrm{L}_\infty(\mathbf{T}) \approx 0.88$.

This means that, if the range clipping could be disregarded, this transformation would not introduce any additional errors after the first encoding/decoding cycle. Unfortunately this is not so. Range clipping for the red, green and blue components takes place in few cases for the input data which range from zero to 255, as it can be seen in Table 5.1.

5.3 Error accumulation for YC_BC_R color transformation in the presence of a range clipping

In order to study the error accumulation in YC_BC_R color transformation the formula **(4.54)**: $\delta^n\vec{\mathbf{X}} = -\delta^n\hat{\vec{\mathbf{X}}} + \mathbf{T}^{-1}\cdot\text{round}\left(\mathbf{T}\cdot\delta^n\hat{\vec{\mathbf{X}}}\right)$, and **(4.63)**: $\delta^{n+1}\hat{\vec{\mathbf{X}}} = -\delta^n\vec{\mathbf{X}} + \text{round}\left(\delta^n\vec{\mathbf{X}}\right)$ must be rewritten in an appropriate way:

$$\begin{bmatrix} \delta^n R \\ \delta^n G \\ \delta^n B \end{bmatrix} = -\begin{bmatrix} \delta^n \hat{R} \\ \delta^n \hat{G} \\ \delta^n \hat{B} \end{bmatrix} + \mathbf{T}^{-1} \cdot \text{round}\left(\mathbf{T} \cdot \begin{bmatrix} \delta^n \hat{R} \\ \delta^n \hat{G} \\ \delta^n \hat{B} \end{bmatrix} \right) \qquad \textbf{(5.08)}$$

$$\begin{bmatrix} \delta^{n+1} \hat{R} \\ \delta^{n+1} \hat{G} \\ \delta^{n+1} \hat{B} \end{bmatrix} = -\begin{bmatrix} \delta^n R \\ \delta^n G \\ \delta^n B \end{bmatrix} + \text{round}\left(\begin{bmatrix} \delta^n R \\ \delta^n G \\ \delta^n B \end{bmatrix} \right) \qquad \textbf{(5.09)}$$

The above formulae represent the dependencies between the increments, thus these equations do not depend on any integer shifts of the origin of the coordinates system. It is comfortable to move the origin of the coordinate system to the points where the clipping errors can originate, and find out how the data evolve while multiple encoding/decoding cycles are performed. As mentioned above the side faces and edges of *RGB* range cube are the areas where such a clipping can happen. The analysis which is done here is similar to the reasoning described in the chapter 4.6. The three dimensional input data range cubes have only three kinds of edges: vertices, edges and corners. For more dimensions the reasoning is much more complex and it is necessary to conduct numerical calculations to analyze the error accumulation.

5.3.1 Side faces of RGB range cube

In order to analyze the range clipping which happens on the side faces of the *RGB* range cube it is convenient to shift the origin of the coordinate system to an original RGB data point lying on the a currently analyzed side face of the X range cuboid. If the critical case (greatest clipping errors) is considered then the original RGB data point (and the origin of the coordinate system) will be located in a vertex of an YC_BC_R elementary cell. From now on the terms "**integer $\boldsymbol{YC_BC_R}$ points**" and "**integer *RGB* points**" should be understood as the points which have integer coordinates in their own frames of reference. The two dimensional example for the critical case (greatest clipping error) is shown in Figure 5.1. As mentioned above the *RGB* frame of reference is assumed to be orthonormal. The black dots on the figure represent "integer *RGB* points". The

RGB range cube turns into a square in two dimensions. The parallelogram is a 2D model of the YC_BC_R elementary cell and all the integer *RGB* points, which are located inside, are rounded to its center. The beginning of the RGB frame of reference (marked on the fig 5.1 with an empty circle) is shifted onto side face of *RGB* range cube, where the considered original RGB data point is located. The empty crossed circle in the center of the parallelogram shows the location of the "integer YC_BC_R point", for which the greatest clipping error will happen. Accordingly, the point marked with the crossed circle has the YC_BC_R coordinates with maximum absolute value 0.5.

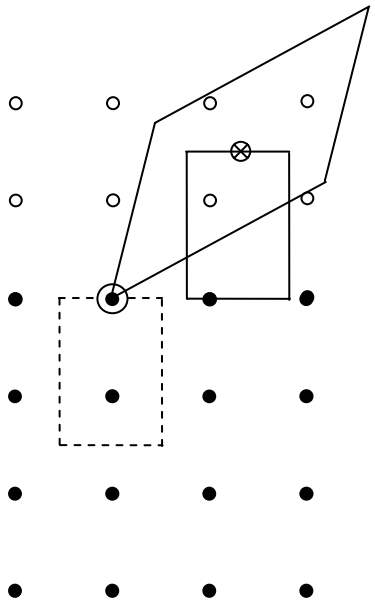

Fig 5.1. Two dimensional case - clipping rectangle.

The center of YC_BC_R elementary cell, which is shown on the Fig. 5.1, sticks out of the *RGB* range cube. This means that after transforming the color points back to the *RGB* system one of the *RGB* coordinates will be clipped. Hence the color point will migrate towards the *RGB* range cube face and then it will find the closest "integer *RGB* point". This "integer *RGB* point" is surely located at the distance of plus or minus half along both the remaining (those not clipped) coordinates. On the two dimensional Fig. 5.1 only one rounded dimension can be seen. Thus the color point has migrated to the adjacent YC_BC_R elementary cell.

The situation presented on the figure 5.1 shows the maximum possible clipping error, which can happen over that *RGB* range cube face. The clipping hyper-cuboid, which turns into a rectangle in two dimensions, has the height equal to the maximum clipping error, and both depth and width equal to one (2×0.5) (see Section 4.6.4). On Fig. 5.1 it is drawn with a solid line. The location of YC_BC_R elementary cell, which is shown on Fig. 5.1, is the worst possible case (as long as the side face is considered) and studying it allows for drawing conclusions about all the other cases. This is also shown on Fig. 5.1. The rectangle, drawn with a dashed line, and its interior show all the possible different locations of the coordinate system origin relative to "integer *RGB* points". When moving the coordinate system origin along the dashed rectangle, the YC_BC_R elementary cell center (integer YC_BC_R point) is drawing the clipping rectangle. In three dimensions the rectangles turn into cuboids.

To find out how many adjacent YC_BC_R elementary cells are to be taken into account (how many secondary migration vectors are to be taken into account) one should find the upper bound for the secondary migration vectors. In order to do this it is necessary to find the upper bounds for YC_BC_R errors: $\delta Y^{\max}$, $\delta C_B^{\max}$ and $\delta C_R^{\max}$. It can be done using matrix $\mathbf{S}_{abs}$ defined in the formula **(4.28)**.

$$\begin{bmatrix} \delta R^{\max} \\ \delta G^{\max} \\ \delta B^{\max} \end{bmatrix} = \mathbf{S}_{abs} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \approx \begin{bmatrix} 1.38 \\ 1.18 \\ 1.59 \end{bmatrix} \qquad \textbf{(5.10)}$$

This results in the following maximum Y coordinates changes:

$$\begin{bmatrix} \delta Y^{\max} \\ \delta C_B^{\max} \\ \delta C_R^{\max} \end{bmatrix} = \mathbf{T}_{abs} \cdot \begin{bmatrix} \delta R^{\max} \\ \delta G^{\max} \\ \delta B^{\max} \end{bmatrix} \approx \begin{bmatrix} 1.12 \\ 1.25 \\ 1.16 \end{bmatrix}, \qquad \textbf{(5.11)}$$

Matrix $\mathbf{T}_{abs}$ is defined in the formula **(4.27)**.

Hence the upper bounds for the secondary migration vector components are as follows:

$$\text{round}\left(\begin{bmatrix} \delta Y^{\max} \\ \delta C_B^{\max} \\ \delta C_R^{\max} \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \textbf{(5.12)}$$

what can be proved in the same way as the proposition 2 (refer to section 4.4).

Equation **(5.12)** implies that a color point can migrate only one YC_BC_R elementary cell further in a single encoding/decoding cycle, thus it is enough to consider the first encoding/decoding cycle YC_BC_R elementary cell and its 26 neighbors as shown of Fig. 5.2.

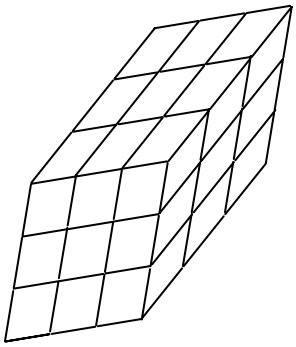

Fig 5.2. The first cycle YC_BC_R cell, located in the center, and its 26 neighbors.

This means that it is enough to take into account 26 secondary migration vectors (refer to Section 4.6.5). When analyzing where the vertices of the clipping cuboid are located it is possible to find out, which adjacent YC_BC_R elementary cells the color point will migrate to, when clipping the one of *RGB* coordinates and rounding the others. The analysis of the very vertices of the clipping cuboid is not sufficient however. It is possible that the clipping cuboid edge crosses one adjacent YC_BC_R elementary cell and ends up in another one. To decide whether the elementary cells, which were found when analyzing the location of the clipping cuboid vertices, are all those to which the color point can migrate, it is necessary to check, whether the first cycle YC_BC_R elementary cell and those adjacent ones form a convex figure. If they do, it will be all right. If they do not, it will be necessary to add those adjacent YC_BC_R elementary cells which will make the

figure convex – another parallelepiped but greater. An example of such a parallelepiped is shown on Fig. 5.3.

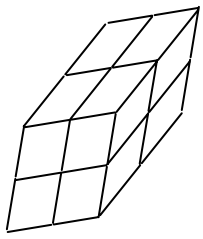

Fig 5.3. The first YC_BC_R cell and its 7 neighbors.

The figure presented above is a subset of the figure presented on Fig. 5.2. In this way the set of secondary migration vectors, which must be taken into account, is reduced to seven elements. At this stage it is time to find out if those adjacent YC_BC_R elementary cells are still far away from the side face of *RGB* range cube. If their centers are distant from the side face by not more then half in *RGB* system, then no more clipping will occur. If it is not so then it will be necessary to consider those adjacent elementary cells more thoroughly.

5.3.1.1 R_{max} and R_{min} face of RGB range cube

For the R_{max} face of *RGB* range cube, the YC_BC_R elementary cell which sticks out in *R* direction most is the one, for which the clipping error is the greatest. Its center has the following coordinates: (0.5, 0.5, 0.5) in YC_BC_R frame of reference. Hence it is spanned on the following YC_BC_R vectors $\mathbf{Y} = (1, 0, 0)$ $\mathbf{C_B} = (0, 1, 0)$ $\mathbf{C_R} = (0, 0, 1)$. The projection of these vectors onto the R_{max} side face is shown on Fig. 5.4.

The vector:

$$\begin{bmatrix} \delta^0 R \\ \delta^0 G \\ \delta^0 B \end{bmatrix} = \mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \approx \begin{bmatrix} 1.38 \\ -0.02 \\ 1.59 \end{bmatrix} \qquad \textbf{(5.13)}$$

represents the *RGB* coordinates of the center of the first YC_BC_R elementary cell relative to the origin of the shifted frame of reference. Instead of the exact $\mathbf{T}^{-1}$ matrix, its approximation **S**,

defined in **(5.04)**, is used. The center of this cell is most distant from R_{max} face of RGB range cube and according to equation **(5.13)** the R component of the YC_BC_R elementary cell center is as follows:

$$\left[\mathbf{S}\cdot\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}\right]_R \approx 1.38 \qquad \textbf{(5.14)}$$

where the subscript R denotes the red component.

The above presented value is equal to the $\delta R^{\max}$ (defined in **(5.10)**), hence it is the YC_BC_R elementary cell center, which is sought.

The equation **(5.09)** for the first cycle is as follows:

$$\begin{bmatrix}\delta^1\hat{R}\\ \delta^1\hat{G}\\ \delta^1\hat{B}\end{bmatrix} = \begin{bmatrix}-\delta^0 R + \text{round}\left(\delta^0 R\right)\\ -\delta^0 G + \text{round}\left(\delta^0 G\right)\\ -\delta^0 B + \text{round}\left(\delta^0 B\right)\end{bmatrix}. \qquad \textbf{(5.15)}$$

Since the R component is clipped and the G and B components are rounded the equation **(5.15)** can be written as follows:

$$\begin{bmatrix}\delta^1\hat{R}\\ \delta^1\hat{G}\\ \delta^1\hat{B}\end{bmatrix} = \begin{bmatrix}-\delta^0 R\\ -\delta^0 G + \text{round}\left(\delta^0 G\right)\\ -\delta^0 B + \text{round}\left(\delta^0 B\right)\end{bmatrix}. \qquad \textbf{(5.16)}$$

Thus **(5.16)** implies:

$$\delta^1\hat{R} \in [-\delta R^{\max}, 0],\ \delta^1\hat{G} \in [-0.5, 0.5] \text{ and } \delta^1\hat{B} \in [-0.5, 0.5]. \qquad \textbf{(5.17)}$$

The conditions **(5.17)** define the lengths of the first encoding/decoding cycle clipping cuboid edges. The secondary migration vector in Y frame of reference:

$$\text{round}\left(\mathbf{T}\cdot\begin{bmatrix}\delta^1\hat{R}\\ \delta^1\hat{G}\\ \delta^1\hat{B}\end{bmatrix}\right) \qquad \textbf{(5.18)}$$

shows, which adjacent YC_BC_R elementary cell center the color point will migrate to in the second encoding/decoding cycle. In order to learn the location of the this second encoding/decoding cycle YC_BC_R elementary cell center relative to the considered original RGB data point it is necessary to add the secondary migration vector to the first encoding/decoding cycle YC_BC_R elementary cell center.

Hence **(5.17)** and **(5.18)** imply that if the red component is clipped and the other two are rounded, then the following secondary migration vectors will have to be considered:

$$\operatorname{round}\left(\mathbf{T}\cdot\begin{bmatrix}-\delta R^{\max}\\-0.5\\-0.5\end{bmatrix}\right)=\begin{bmatrix}-1\\0\\0\end{bmatrix} \qquad \textbf{(5.19)}$$

$$\operatorname{round}\left(\mathbf{T}\cdot\begin{bmatrix}-\delta R^{\max}\\0.5\\0.5\end{bmatrix}\right)=\begin{bmatrix}0\\0\\-1\end{bmatrix} \qquad \textbf{(5.20)}$$

$$\operatorname{round}\left(\mathbf{T}\cdot\begin{bmatrix}-\delta R^{\max}\\0.5\\-0.5\end{bmatrix}\right)=\begin{bmatrix}0\\0\\-1\end{bmatrix} \qquad \textbf{(5.21)}$$

$$\operatorname{round}\left(\mathbf{T}\cdot\begin{bmatrix}-\delta R^{\max}\\-0.5\\0.5\end{bmatrix}\right)=\begin{bmatrix}-1\\1\\0\end{bmatrix} \qquad \textbf{(5.22)}$$

Accordingly the second encoding/decoding cycle YC_BC_R elementary cells, in which the four vertices of clipping cuboid are situated, have their centers located in the shifted YC_BC_R coordinate system in the following points:

From **(5.19)** $\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}-1\\0\\0\end{bmatrix}=\begin{bmatrix}-0.5\\0.5\\0.5\end{bmatrix}$, thus its R component: $\left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\0.5\\0.5\end{bmatrix}\right]_R\approx 0.22$ **(5.23)**

From **(5.20)** $\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}0\\0\\-1\end{bmatrix}=\begin{bmatrix}0.5\\0.5\\-0.5\end{bmatrix}$, thus its R component: $\left[\mathbf{S}\cdot\begin{bmatrix}0.5\\0.5\\-0.5\end{bmatrix}\right]_R\approx -0.22$ **(5.24)**

From **(5.22)** $\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}-1\\1\\0\end{bmatrix}=\begin{bmatrix}-0.5\\1.5\\0.5\end{bmatrix}$ thus its R component: $\left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\1.5\\0.5\end{bmatrix}\right]_R\approx 0.22$ **(5.25)**

The first encoding/decoding cycle YC_BC_R elementary cell and those, to which the color point can migrate in the second encoding/decoding cycle (the secondary migration vectors **(5.19)**, **(5.20)** and **(5.22)**), do not constitute a convex figure. To create a convex figure the following secondary migration vectors must be included:

$$\begin{bmatrix}-1\\0\\-1\end{bmatrix},\begin{bmatrix}0\\1\\-1\end{bmatrix},\begin{bmatrix}-1\\1\\-1\end{bmatrix},\begin{bmatrix}0\\1\\0\end{bmatrix}. \qquad \textbf{(5.26)}$$

The centers of these additional second encoding/decoding cycle YC_BC_R elementary cells have following YC_BC_R coordinates:

$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}-1\\0\\-1\end{bmatrix}=\begin{bmatrix}-0.5\\0.5\\-0.5\end{bmatrix}$ thus its R component: $\left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\0.5\\-0.5\end{bmatrix}\right]_R\approx -1.38$ **(5.27)**

$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}0\\1\\-1\end{bmatrix}=\begin{bmatrix}0.5\\1.5\\-0.5\end{bmatrix}$ thus its R component: $\left[\mathbf{S}\cdot\begin{bmatrix}0.5\\1.5\\-0.5\end{bmatrix}\right]_R\approx -0.22$ **(5.28)**

$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}-1\\1\\-1\end{bmatrix}=\begin{bmatrix}-0.5\\1.5\\-0.5\end{bmatrix}$ thus its R component: $\left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\1.5\\-0.5\end{bmatrix}\right]_R\approx -1.38$ **(5.29)**

$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}=\begin{bmatrix}0.5\\1.5\\0.5\end{bmatrix}$ thus its R component: $\left[\mathbf{S}\cdot\begin{bmatrix}0.5\\1.5\\0.5\end{bmatrix}\right]_R\approx 1.38$ **(5.30)**

The negative values of the *R* components and those positive which are not greater then half imply that no further range clipping will take place in these cases. Thus in the following cases **(5.23)**, **(5.24)**, **(5.25)**, **(5.27)**, **(5.28)** and **(5.29)**, the color point have either entered the *RGB* range cube, or approached its side face at the distance smaller then half for *R* component. The value of *R* component, which is positive and greater then half, means that another range clipping may take place. Thus the case **(5.30)** needs more attention. When comparing the values of *R* components of the first cell **(5.14)** and the last adjacent cell **(5.30)** one can see no difference. This suggests that the centers of both the YC_BC_R elementary cells are equally distant from the R_{max} face of the *RGB* range cube. Of course, when reasoning is conducted with the rounded real numbers, only the "smaller-greater" relation between the considered numbers can be deduced with certainty. To check, if the considered cells are really equally distant from the R_{max} face of the *RGB* range cube, it is necessary to take a closer look at the secondary migration vector. Its components in the YC_BC_R coordinates system are following:

$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \textbf{(5.31)}$$

(see **(5.26)**; actually this is the base vector $\mathbf{C_B}$).

Let us recall the definition **(5.04)**

$$\mathbf{T}^{-1} \approx \mathbf{S} = \frac{1}{256}\begin{bmatrix} 298.082 & 0 & 408.583 \\ 298.082 & -100.291 & -208.120 \\ 298.082 & 516.411 & 0 \end{bmatrix} \qquad \textbf{(5.32)}$$

Hence:

$$\mathbf{S}\cdot\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{256}\cdot\begin{bmatrix} 298.082 & 0 & 408.583 \\ 298.082 & -100.291 & -208.120 \\ 298.082 & 516.411 & 0 \end{bmatrix}\cdot\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \frac{1}{256}\cdot\begin{bmatrix} 0 \\ -100.291 \\ 516.411 \end{bmatrix} \qquad \textbf{(5.33)}$$

Thus the R component of the secondary migration vector is exactly zero. This implies that both the first YC_BC_R elementary cell center and the center of the considered adjacent one are equally distant from R_{max} side face of the RGB range cube. Hence due to range clipping of the R component, the migration of the color point along the C_B axis of the YC_BC_R coordinate system might be possible. To find out, if such a migration may occur, it is necessary to check, if the clipping cuboid of the first encoding/decoding cycle YC_BC_R elementary cell and the considered adjacent YC_BC_R elementary cell have any common part. The best way to check it, is to project both these figures onto the R_{max} side face of the RGB range cube.

The considered clipping cuboid vertices have following coordinates in the RGB frame of reference (their projections onto the R_{max} side face are marked with black squares on Fig. 5.4):

1. $(0, G_1, B_1)$
2. $(0, G_1, B_2)$
3. $(0, G_2, B_1)$
4. $(0, G_2, B_2)$
5. $(\delta R^{\max}, G_1, B_1)$
6. $(\delta R^{\max}, G_1, B_2)$
7. $(\delta R^{\max}, G_2, B_1)$
8. $(\delta R^{\max}, G_2, B_2)$

where:

$$G_1 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right]_G - 0.5 \approx -0.52 \qquad \textbf{(5.19)}$$

$$G_2 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right]_G + 0.5 \approx 0.48 \qquad \textbf{(5.20)}$$

$$B_1 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right]_B - 0.5 \approx 1.09 \qquad \textbf{(5.21)}$$

$$B_2 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right]_B + 0.5 \approx 2.09 \qquad \textbf{(5.22)}$$

$$\delta R^{\max} \approx 1.38 \qquad \textbf{(5.23)}$$

The projection of both these figures onto the R_{max} side face of the *RGB* range cube is presented on fig. 5.4. The small black squares indicate the projections of the clipping cuobid vertices onto the *RGB* range cube face. The small black squares which are rotated indicate the projections of the considered second encoding/decoding cycle YC_BC_R elementary cell vertices onto the *RGB* range cube face. The two numbers located in the vicinity of each vertex show its *G* and *B* coordinates. The arrows drawn there represent the base vectors of the shifted YC_BC_R coordinate system, on which the first YC_BC_R elementary cell is spanned.

Vector $\mathbf{C_B}$ (and also C_B axis, of course) belongs to the R_{max} side faces of the *RGB* range cube thus the arrow drawn on the Fig. 5.4 is not its projection but this vector itself. As it can be seen on the Fig. 5.4, the clipping cuboid of the first encoding/decoding cycle YC_BC_R elementary cell and the considered adjacent YC_BC_R elementary cell have a common part, which is denoted as **A**. Area **B** corresponds to the area **A**, which is moved by the vector $\mathbf{C_B}$. Areas **C** and **D** correspond to the area **A**, which is moved according to *RGB* frame of reference by the vectors (0, -1, 2) and (0, 0, 2) respectively. Thus the color point might migrate from the first YC_BC_R elementary cell to the considered adjacent one if an "integer *RGB* point" were located on the R_{max} side face of the *RGB* range cube in the area belonging to the mentioned above common part **A**. If the "integer *RGB* point" belongs to the area **A**, then the other "integer *RGB* points" will belong to the areas **C**, **D** and others moved by integer number along *G* and/or *B* axis – but not to the area **B**. Thus it is not possible to migrate along C_B axis for more cycles then one. It is still

possible that the color point will not stay in the considered second YC_BC_R elementary cell and will migrate to another one, but all those to which it can migrate, are close enough to the R_{max} face of the *RGB* range cube to make the range clipping impossible. It is also impossible for the color point to return to the first YC_BC_R elementary cell because the clipping cuboid of the second YC_BC_R elementary cell has no common part with the first YC_BC_R elementary cell.

Hence it is proved that range clipping over the R_{max} side face of the *RGB* range cube may cause error accumulation up to the third encoding/decoding cycle. The reasoning for the R_{min} face of the *RGB* range cube is identical. The only difference is that the signs are opposite.

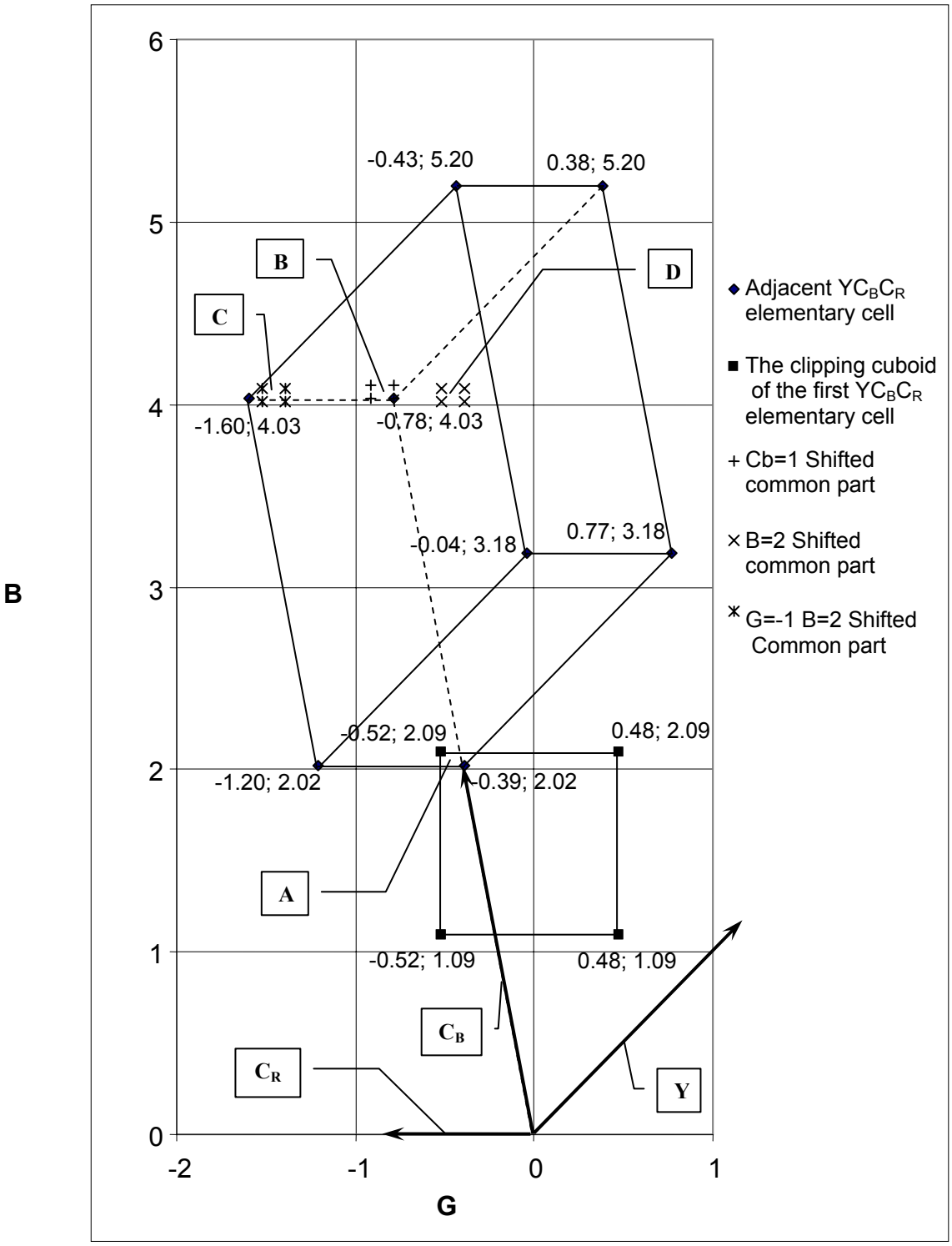

Fig 5.4. Verifying the additional YCbCr cell over the R_{max} face.

5.3.1.2 G_{max} and G_{min} face of RGB range cube

For the G_{max} face of the *RGB* range cube, the YC_BC_R elementary cell, which sticks out most is the one, the center of which has the following YC_BC_R coordinates (0.5, -0.5, -0.5). Hence it is spanned on the following YC_BC_R vectors: $\mathbf{Y} = (1, 0, 0)$, $-\mathbf{C_B} = (0, -1, 0)$, $-\mathbf{C_R} = (0, 0, -1)$.

The vector:

$$\begin{bmatrix} \delta^0 R \\ \delta^0 G \\ \delta^0 B \end{bmatrix} = \mathbf{S} \cdot \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} \approx \begin{bmatrix} -0.22 \\ 1.18 \\ -0.43 \end{bmatrix} \qquad \textbf{(5.43)}$$

represents the *RGB* coordinates of the center of the first YC_BC_R elementary cell relative to the origin of the shifted frame of reference. The center of this cell is most distant from G_{max} face of the *RGB* range cube and according to the equation **(5.43)** the *G* component of the YC_BC_R elementary cell center is as follows:

$$\left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} \right]_G \approx 1.18 \qquad \textbf{(5.44)}$$

where the subscript *G* denotes the green component.

The above presented value is equal to $\delta G^{\max} \approx 1.18$, which is defined in **(5.10)**.

The equation **(5.09)** for the first cycle in this case is as follows:

$$\begin{bmatrix} \delta^1 \hat{R} \\ \delta^1 \hat{G} \\ \delta^1 \hat{B} \end{bmatrix} = \begin{bmatrix} -\delta^0 R & + \text{round}\left(\delta^0 R\right) \\ & -\delta^0 G \\ -\delta^0 B & + \text{round}\left(\delta^0 B\right) \end{bmatrix}. \qquad \textbf{(5.45)}$$

In the above equation the fact, that the *G* component is clipped and the *R* and *B* components are rounded, is exploited.

Thus

$$\delta^1 \hat{R} \in [-0.5, 0.5],\ \delta^1 \hat{G} \in [-\delta G^{\max}, 0] \text{ and } \delta^1 \hat{B} \in [-0.5, 0.5]. \qquad \textbf{(5.46)}$$

Hence the following secondary migration vectors must be considered:

$$\text{round}\left(\mathbf{T}\cdot\begin{bmatrix}-0.5\\-\delta G^{\max}\\-0.5\end{bmatrix}\right)=\begin{bmatrix}-1\\0\\0\end{bmatrix} \tag{5.47}$$

$$\text{round}\left(\mathbf{T}\cdot\begin{bmatrix}0.5\\-\delta G^{\max}\\0.5\end{bmatrix}\right)=\begin{bmatrix}0\\0\\1\end{bmatrix} \tag{5.48}$$

$$\text{round}\left(\mathbf{T}\cdot\begin{bmatrix}-0.5\\-\delta G^{\max}\\0.5\end{bmatrix}\right)=\begin{bmatrix}-1\\1\\0\end{bmatrix} \tag{5.49}$$

$$\text{round}\left(\mathbf{T}\cdot\begin{bmatrix}0.5\\-\delta G^{\max}\\-0.5\end{bmatrix}\right)=\begin{bmatrix}-1\\0\\1\end{bmatrix} \tag{5.50}$$

The adjacent YC_BC_R elementary cells, in which the four vertices of the clipping cuboid are situated, have their centers located, according to YC_BC_R coordinate system, in the following points:

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}-1\\0\\0\end{bmatrix}=\begin{bmatrix}-0.5\\-0.5\\-0.5\end{bmatrix}\quad\text{thus its } G \text{ component: }\left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\-0.5\\-0.5\end{bmatrix}\right]_G\approx 0.02 \tag{5.51}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\0\\1\end{bmatrix}=\begin{bmatrix}0.5\\-0.5\\0.5\end{bmatrix}\quad\text{thus its } G \text{ component: }\left[\mathbf{S}\cdot\begin{bmatrix}0.5\\-0.5\\0.5\end{bmatrix}\right]_G\approx 0.37 \tag{5.52}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}-1\\1\\0\end{bmatrix}=\begin{bmatrix}-0.5\\0.5\\-0.5\end{bmatrix}\quad\text{thus its } G \text{ component: }\left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\0.5\\-0.5\end{bmatrix}\right]_G\approx -0.37 \tag{5.53}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}-1\\0\\1\end{bmatrix}=\begin{bmatrix}-0.5\\-0.5\\0.5\end{bmatrix}\quad\text{thus its } G \text{ component: }\left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\-0.5\\0.5\end{bmatrix}\right]_G\approx -0.79 \tag{5.54}$$

The first YC_BC_R elementary cell and those, to which the color point will migrate with the secondary migration vectors **(5.47)**, **(5.48)**, **(5.49)** and **(5.50)**, do not constitute a convex figure. To create a convex figure the following secondary migration vectors must be included:

$$\begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}-1\\1\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}. \qquad \textbf{(5.55)}$$

The centers of these additional YC_BC_R elementary cells have the following YC_BC_R coordinates:

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\1\end{bmatrix}=\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix} \quad \text{thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}\right]_G \approx -0.02 \qquad \textbf{(5.56)}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}-1\\1\\1\end{bmatrix}=\begin{bmatrix}-0.5\\0.5\\0.5\end{bmatrix} \quad \text{thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\0.5\\0.5\end{bmatrix}\right]_G \approx -1.18 \qquad \textbf{(5.57)}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}=\begin{bmatrix}0.5\\0.5\\-0.5\end{bmatrix} \quad \text{thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\0.5\\-0.5\end{bmatrix}\right]_G \approx 0.79 \qquad \textbf{(5.58)}$$

The negative values of the above presented G components and those positive which are not greater then half imply that no further range clipping will take place in these cases. Thus in the cases **(5.51)**, **(5.52)**, **(5.53)**, **(5.54)**, **(5.55)**, **(5.56)** and **(5.57)**, the color point have either entered the *RGB* range cube, or approached its side face at the distance smaller then half for G component. The value of G component, which is positive and greater then half, means that another range clipping may take place. Thus for the case **(5.58)** the further analysis is necessary. Fig. 5.5 represents the projection of the clipping cuboid of the first YC_BC_R elementary cell and the projection of the second YC_BC_R elementary cell onto the G_{max} side face of the *RGB* range cube for the case **(5.62)**. The vectors **Y**, $-\mathbf{C_B}$ and $-\mathbf{C_R}$ span the first YC_BC_R elementary cell. The two numbers located in the vicinity of each vertex show its R and B coordinates.

The expression $\mathbf{S}\cdot\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}$ is explained in **(5.43)**.

The vertices of the clipping cuboid have the following *RGB* coordinates in the shifted frame of reference (their projections onto the G_{max} side face are marked with rotated black squares in Fig. 5.5):

1. $(R_1, 0, B_1)$
2. $(R_1, 0, B_2)$
3. $(R_2, 0, B_1)$
4. $(R_2, 0, B_2)$
5. $(R_1, \delta G^{\max}, B_1)$
6. $(R_1, \delta G^{\max}, B_2)$
7. $(R_2, \delta G^{\max}, B_1)$
8. $(R_2, \delta G^{\max}, B_2)$

where:

$$R_1 = \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}\right]_R - 0.5 \approx -0.72 \qquad \textbf{(5.59)}$$

$$R_2 = \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}\right]_R + 0.5 \approx 0.28 \qquad \textbf{(5.60)}$$

$$B_1 = \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}\right]_B - 0.5 \approx -0.93 \qquad \textbf{(5.61)}$$

$$B_2 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \end{bmatrix} \right]_B + 0.5 \approx 0.07 \qquad \textbf{(5.62)}$$

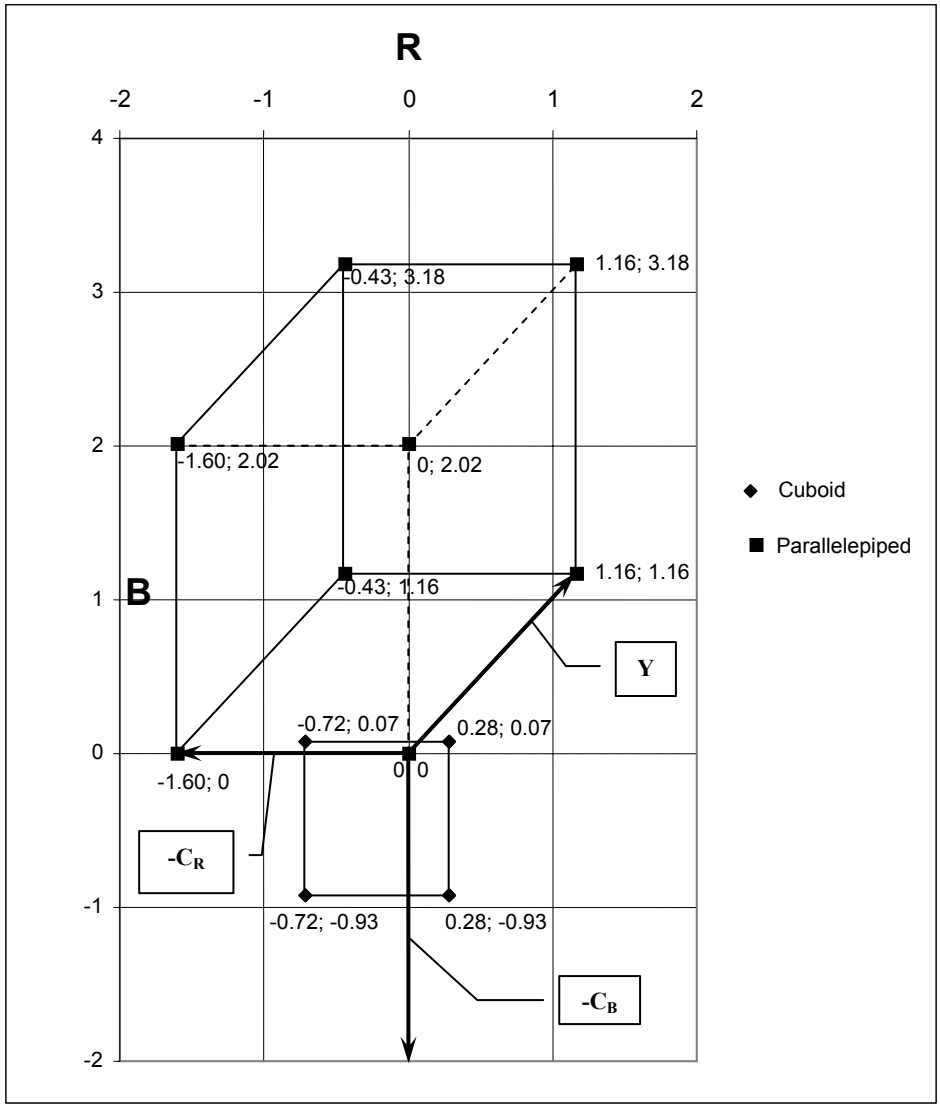

Fig 5.5. Verifying the additional YCbCr cell over the G_{max} face.

One can see that these two figures have a common part. This means that the analysis of the second encoding/decoding cycle is necessary.

Let us define **(**case **(5.58))**:

$$\delta G_{2'nd\ Clip} = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \right]_G \tag{5.63}$$

The equation **(5.09)** for the second cycle in this case is as follows:

$$\begin{bmatrix} \delta^2\hat{R} \\ \delta^2\hat{G} \\ \delta^2\hat{B} \end{bmatrix} = \begin{bmatrix} -\delta^1 R & +\text{round}\left(\delta^1 R\right) \\ & -\delta^1 G \\ -\delta^1 B & +\text{round}\left(\delta^1 B\right) \end{bmatrix}. \tag{5.64}$$

Thus:

$$\delta^2\hat{R} \in [-0.5, 0.5],\ \delta^2\hat{G} \in [-\delta G_{2'nd\ Clip}, 0] \text{ and } \delta^2\hat{B} \in [-0.5, 0.5]. \tag{5.65}$$

The secondary migration vector for the second encoding decoding cycle in YC_BC_R frame of reference is as follows:

$$\text{round}\left(\mathbf{T} \cdot \begin{bmatrix} \delta^2\hat{R} \\ \delta^2\hat{G} \\ \delta^2\hat{B} \end{bmatrix} \right) \tag{5.66}$$

To find out what is happening in the next encoding/decoding cycle it is necessary to consider the following secondary migration vectors:

$$\text{round}\left(\mathbf{T} \cdot \begin{bmatrix} -0.5 \\ -\delta G_{2'nd\ Clip} \\ -0.5 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix} \tag{5.67}$$

$$\text{round}\left(\mathbf{T} \cdot \begin{bmatrix} 0.5 \\ -\delta G_{2'nd\ Clip} \\ 0.5 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.68}$$

$$\text{round}\left(\mathbf{T} \cdot \begin{bmatrix} -0.5 \\ -\delta G_{2'nd\ Clip} \\ 0.5 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{5.69}$$

$$\text{round}\left(\mathbf{T} \cdot \begin{bmatrix} 0.5 \\ -\delta G_{2'nd\ Clip} \\ -0.5 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.70}$$

The vector:

$$\mathbf{S}\cdot\begin{bmatrix}0.5\\0.5\\-0.5\end{bmatrix}=\mathbf{S}\cdot\left[\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}\right] \qquad \textbf{(5.71)}$$

represents the *RGB* coordinates of the center of the considered second YC_BC_R elementary cell relative to the origin of the shifted frame of reference.

The adjacent YC_BC_R elementary cells, in which the four vertices of clipping cuboid are situated, have their centers located, according to YC_BC_R coordinate system, in the following points:

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}-1\\0\\0\end{bmatrix}=\begin{bmatrix}-0.5\\0.5\\-0.5\end{bmatrix} \text{ thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\0.5\\-0.5\end{bmatrix}\right]_G\approx -0.37 \qquad \textbf{(5.72)}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}=\begin{bmatrix}0.5\\1.5\\-0.5\end{bmatrix} \text{ thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\1.5\\-0.5\end{bmatrix}\right]_G\approx 0.40 \qquad \textbf{(5.73)}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}0\\0\\1\end{bmatrix}=\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix} \text{ thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}\right]_G\approx -0.02 \qquad \textbf{(5.74)}$$

The second YC_BC_R elementary cell and those, to which the color point will migrate with the secondary migration vectors **(5.67)**, **(5.96)** and **(5.70)**, do not constitute a convex figure. In order to create a convex figure the following secondary migration vectors must be included:

$$\begin{bmatrix}-1\\1\\0\end{bmatrix},\begin{bmatrix}-1\\0\\1\end{bmatrix},\begin{bmatrix}0\\1\\1\end{bmatrix},\begin{bmatrix}-1\\1\\1\end{bmatrix}. \qquad \textbf{(5.75)}$$

The centers of these additional YC_BC_R elementary cells have the following YC_BC_R coordinates:

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}-1\\1\\0\end{bmatrix}=\begin{bmatrix}-0.5\\1.5\\-0.5\end{bmatrix} \text{ thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\1.5\\-0.5\end{bmatrix}\right]_G\approx -0.76 \qquad \textbf{(5.76)}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}-1\\0\\1\end{bmatrix}=\begin{bmatrix}-0.5\\0.5\\0.5\end{bmatrix}\text{ thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\0.5\\0.5\end{bmatrix}\right]_G\approx -1.18 \qquad \textbf{(5.77)}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}0\\1\\1\end{bmatrix}=\begin{bmatrix}0.5\\1.5\\0.5\end{bmatrix}\text{ thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\1.5\\0.5\end{bmatrix}\right]_G\approx -0.41 \qquad \textbf{(5.78)}$$

$$\begin{bmatrix}0.5\\-0.5\\-0.5\end{bmatrix}+\begin{bmatrix}0\\1\\0\end{bmatrix}+\begin{bmatrix}-1\\1\\1\end{bmatrix}=\begin{bmatrix}-0.5\\1.5\\0.5\end{bmatrix}\text{ thus its } G \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\1.5\\0.5\end{bmatrix}\right]_G\approx -1.58 \qquad \textbf{(5.79)}$$

When exploring the cases **(5.72)**, **(5.73)**, **(5.74)**, **(5.76)**, **(5.77)**, **(5.78)** and **(5.79)** it is clear that no further range clipping is going to take place. All the centers of the third YC_BC_R elementary cells are located either inside the *RGB* range cube or outside at the distance smaller then half along the *G* axis. Hence it is proved that range clipping over the G_{max} side face of the *RGB* range cube may cause error accumulation up to the third encoding/decoding cycle. The reasoning for the G_{min} face of the *RGB* range cube is identical. The only difference is that the signs are opposite. The vertices of the clipping cuboid have the following *RGB* coordinates in the shifted frame of reference:

1. $(R_1, 0, B_1)$
2. $(R_1, 0, B_2)$
3. $(R_2, 0, B_1)$
4. $(R_2, 0, B_2)$
5. $(R_1, \delta G_{2'nd\,Clip}, B_1)$
6. $(R_1, \delta G_{2'nd\,Clip}, B_2)$
7. $(R_2, \delta G_{2'nd\,Clip}, B_1)$
8. $(R_2, \delta G_{2'nd\,Clip}, B_2)$

where:

$$R_1 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \right]_R - 0.5 \approx -0.72 \qquad \textbf{(5.80)}$$

$$R_2 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \right]_R + 0.5 \approx 0.28 \qquad \textbf{(5.81)}$$

$$B_1 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \right]_B - 0.5 \approx 1.09 \qquad \textbf{(5.82)}$$

$$B_2 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ -0.5 \end{bmatrix} \right]_B + 0.5 \approx 2.09 \qquad \textbf{(5.83)}$$

5.3.1.3 B_{max} and B_{min} face of RGB range cube

As long as the B_{max} face of *RGB* range cube is considered, the YC_BC_R elementary cell which sticks out most is the one, the center of which has the following YC_BC_R coordinates (0.5, 0.5, 0.5). Hence it is spanned on the following YC_BC_R vectors $\mathbf{Y} = (1, 0, 0)$ $\mathbf{C_B} = (0, 1, 0)$ $\mathbf{C_R} = (0, 0, 1)$.

The vector:

$$\begin{bmatrix} \delta^0 R \\ \delta^0 G \\ \delta^0 B \end{bmatrix} = \mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \approx \begin{bmatrix} 1.38 \\ -0.02 \\ 1.59 \end{bmatrix} \qquad \textbf{(5.84)}$$

represents the *RGB* coordinates of the center of the first YC_BC_R elementary cell relative to the origin of the shifted frame of reference. The center of this cell is most distant from B_{max} face of *RGB* range cube and according to the equation **(5.84)** the *B* component of the YC_BC_R elementary cell center is as follows:

$$\left[\mathbf{S}\cdot\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}\right]_B \approx 1.59 \tag{5.85}$$

where the subscript B denotes the blue component.

The above presented value is equal to $\delta B^{\max} \approx 1.59$, which was defined in **(5.10)**.

The equation **(5.09)** for the first cycle in this case is as follows:

$$\begin{bmatrix}\delta^1\hat{R}\\ \delta^1\hat{G}\\ \delta^1\hat{B}\end{bmatrix} = \begin{bmatrix}-\delta^0 R & +\operatorname{round}\left(\delta^0 R\right)\\ -\delta^0 G & +\operatorname{round}\left(\delta^0 G\right)\\ & -\delta^0 B\end{bmatrix}. \tag{5.86}$$

In the above equation the fact, that the B component is clipped and the R and G components are rounded, is exploited.

Thus

$$\delta^1\hat{R} \in [-0.5, 0.5],\ \delta^1\hat{G} \in [-0.5, 0.5] \text{ and } \delta^1\hat{B} \in [-\delta B^{\max}, 0]. \tag{5.87}$$

Hence the following secondary migration vectors must be considered:

$$\operatorname{round}\left(\mathbf{T}\cdot\begin{bmatrix}-0.5\\-0.5\\-\delta B^{\max}\end{bmatrix}\right) = \begin{bmatrix}-1\\0\\0\end{bmatrix} \tag{5.88}$$

$$\operatorname{round}\left(\mathbf{T}\cdot\begin{bmatrix}0.5\\0.5\\-\delta B^{\max}\end{bmatrix}\right) = \begin{bmatrix}0\\-1\\0\end{bmatrix} \tag{5.89}$$

$$\operatorname{round}\left(\mathbf{T}\cdot\begin{bmatrix}-0.5\\0.5\\-\delta B^{\max}\end{bmatrix}\right) = \begin{bmatrix}0\\-1\\0\end{bmatrix} \tag{5.90}$$

$$\operatorname{round}\left(\mathbf{T}\cdot\begin{bmatrix}0.5\\-0.5\\-\delta B^{\max}\end{bmatrix}\right) = \begin{bmatrix}0\\-1\\1\end{bmatrix} \tag{5.91}$$

The adjacent YC_BC_R elementary cells, in which the four vertices of clipping cuboid are situated, have their centers located, according to the shifted YC_BC_R coordinate system, in the following points:

$$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}-1\\0\\0\end{bmatrix}=\begin{bmatrix}-0.5\\0.5\\0.5\end{bmatrix}\ \text{thus its } B \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\0.5\\0.5\end{bmatrix}\right]_B\approx 0.43 \qquad \textbf{(5.92)}$$

$$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}0\\-1\\0\end{bmatrix}=\begin{bmatrix}0.5\\-0.5\\0.5\end{bmatrix}\ \text{thus its } B \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\-0.5\\0.5\end{bmatrix}\right]_B\approx -0.43 \qquad \textbf{(5.93)}$$

$$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}0\\-1\\1\end{bmatrix}=\begin{bmatrix}0.5\\-0.5\\1.5\end{bmatrix}\ \text{thus its } B \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\-0.5\\1.5\end{bmatrix}\right]_B\approx -0.43 \qquad \textbf{(5.94)}$$

The first YC_BC_R elementary cell and those, to which the color point will migrate with the secondary migration vectors defined in **(5.92)**, **(5.93)** and **(5.95)**, do not constitute a convex figure. To create a convex figure the following secondary migration vectors must be included:

$$\begin{bmatrix}-1\\-1\\1\end{bmatrix},\begin{bmatrix}-1\\-1\\0\end{bmatrix},\begin{bmatrix}-1\\0\\1\end{bmatrix},\begin{bmatrix}0\\0\\1\end{bmatrix}. \qquad \textbf{(5.95)}$$

The centers of these additional YC_BC_R elementary cells have following YC_BC_R coordinates:

$$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}-1\\-1\\1\end{bmatrix}=\begin{bmatrix}-0.5\\-0.5\\1.5\end{bmatrix}\ \text{thus its } B \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\-0.5\\1.5\end{bmatrix}\right]_B\approx -1.59 \qquad \textbf{(5.96)}$$

$$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}-1\\-1\\0\end{bmatrix}=\begin{bmatrix}-0.5\\-0.5\\0.5\end{bmatrix}\ \text{thus its } B \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\-0.5\\0.5\end{bmatrix}\right]_R\approx -1.59 \qquad \textbf{(5.97)}$$

$$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}-1\\0\\1\end{bmatrix}=\begin{bmatrix}-0.5\\0.5\\1.5\end{bmatrix}\ \text{thus its } B \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}-0.5\\0.5\\1.5\end{bmatrix}\right]_B\approx 0.43 \qquad \textbf{(5.98)}$$

$$\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}+\begin{bmatrix}0\\0\\1\end{bmatrix}=\begin{bmatrix}0.5\\0.5\\1.5\end{bmatrix}\qquad \text{thus its } B \text{ component: } \left[\mathbf{S}\cdot\begin{bmatrix}0.5\\0.5\\1.5\end{bmatrix}\right]_B\approx 1.59 \qquad \textbf{(5.99)}$$

The negative values of the above presented B components and those positive which are not greater then half imply that no further range clipping will take place in these cases. Thus in the cases **(5.92)**, **(5.93)**, **(5.94)**, **(5.96)**, **(5.97)**, and **(5.98)**, the color point have either entered the *RGB* range cube, or approached its side face at the distance smaller then half for B component. The value of B component, which is positive and greater then half, means that another range clipping may take place. The center of the first YC_BC_R elementary cell is as distant from the B_{max} face of the *RGB* range cube, as the center of the second YC_BC_R elementary cell in the case **(5.99)**. This can be proved as it was done in a similar case for the R_{max} face of the *RGB* range cube in the chapter 5.1.1. Fig. 5.6 represents the projection of the clipping cuboid of the first YC_BC_R elementary cell and the projection of the second YC_BC_R elementary cell onto the B_{max} side face of the *RGB* range cube for the case **(5.99)**. The base vectors **Y**, $\mathbf{C_B}$ and $\mathbf{C_R}$ span the first YC_BC_R elementary cell.

The vertices of the clipping cuboid have the following *RGB* coordinates in the shifted frame of reference (their projections onto the B_{max} side face are marked with black squares in Fig. 5.6):

1. $(R_1, G_1, 0)$
2. $(R_1, G_2, 0)$
3. $(R_2, G_1, 0)$
4. $(R_2, G_2, 0)$
5. $(R_1, G_1, \delta B^{\max})$
6. $(R_1, G_2, \delta B^{\max})$
7. $(R_2, G_1, \delta B^{\max})$
8. $(R_2, G_2, \delta B^{\max})$

where:

$$R_1 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right]_R - 0.5 \approx 0.88 \qquad \textbf{(5.100)}$$

$$R_2 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right]_R + 0.5 \approx 1.88 \qquad \textbf{(5.101)}$$

$$G_1 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right]_G - 0.5 \approx -0.52 \qquad \textbf{(5.102)}$$

$$G_2 = \left[\mathbf{S} \cdot \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \right]_G + 0.5 \approx 0.48 \qquad \textbf{(5.103)}$$

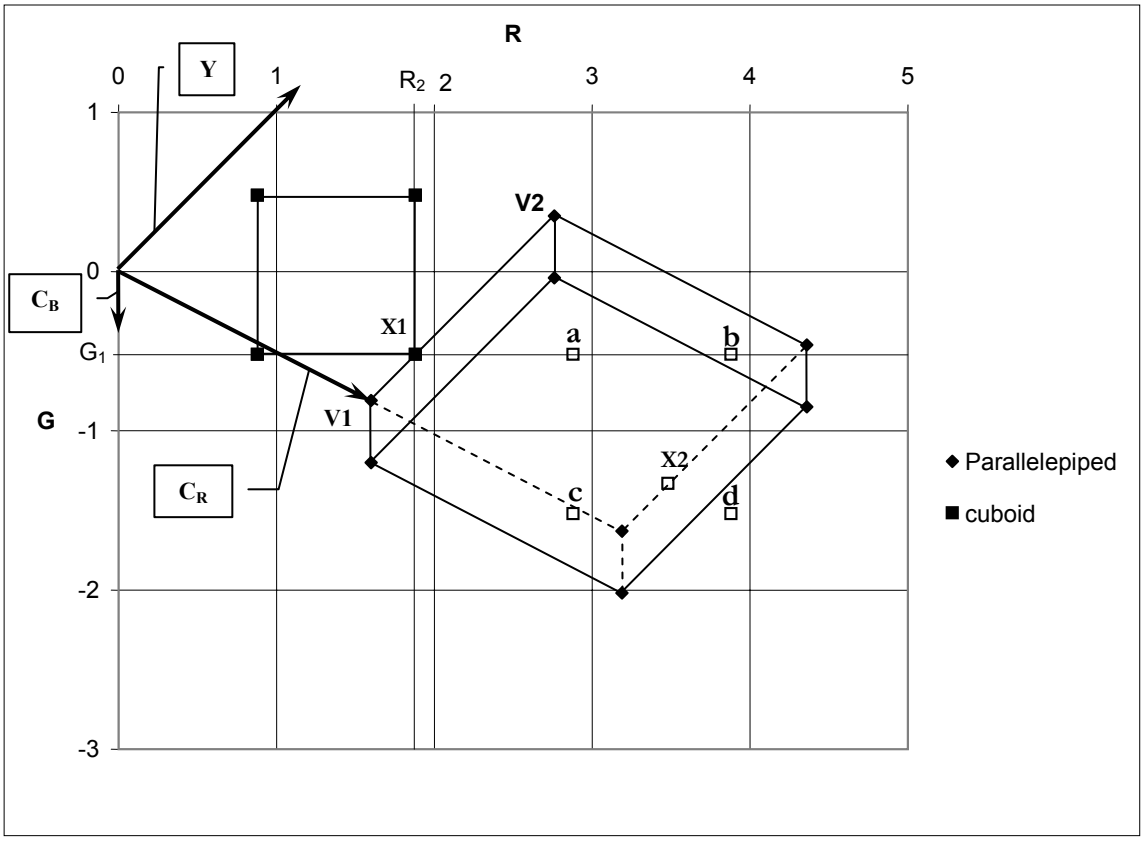

Fig 5.6. Verifying the additional YCbCr cell over the B_{max} face.

The vertex of the first YC_BC_R elementary cell clipping cuboid, having the following coordinates: (R_2, G_1, 0) in the shifted *RGB* frame of reference is denoted as **X1**. It may belong to the second YC_BC_R elementary cell. Hence around this point the common part of these two figures may be located. This vertex moved by the $\mathbf{C_R}$ vector is denoted as **X2**. The points **a**, **b**, **c** and **d** are moved relative the **X1** point, according to the *RGB* frame of reference, by the following vectors (1, 0, 0), (2, 0, 0), (1, -1, 0) and (2, -1, 0) respectively.

It is necessary to find out weather the point **X1** is located inside, outside or on the edge of the second YC_BC_R elementary cell.

Let us consider an edge of the second YC_BC_R elementary cell, which is parallel to the **Y** vector. On its end the vertices **V1** and **V2** are located. In order to find the point, in which this edge crosses the following plane

$$R = R_2, \tag{5.104}$$

the equation **(5.105)** must be solved

$$[\mathbf{C_R} + y \cdot \mathbf{Y}]_R = R_2, \tag{5.105}$$

where y is the unknown variable.

The equation **(5.105)** can be rewritten in the following way:

$$\left[\mathbf{S} \cdot \left[\begin{pmatrix}0\\0\\1\end{pmatrix} + y \cdot \begin{pmatrix}1\\0\\0\end{pmatrix}\right]\right]_R = R_2. \tag{5.106}$$

Hence:

$$y = \frac{-s_{13} + R_2}{s_{11}} \approx 0.24. \tag{5.107}$$

The value of y belongs to the interval [0, 1], hence using it one can find the point, which is sought:

$$[\mathbf{C_R} + y \cdot \mathbf{Y}]_G \approx -0.53. \tag{5.108}$$

Thus the approximate values of the *R* and *G* coordinates of this point are as follows:

$$R \approx 1{,}88 \text{ and } G \approx -0.53$$

This point is denoted on the fig. 5.7 as **Y1**.

Let us find the point, in which this edge crosses the following plane

$$G = G_1, \tag{5.109}$$

The equation **(5.110)** must be solved in order to find this point.

$$[\mathbf{C_R} + y \cdot \mathbf{Y}]_G = G_1 \tag{5.110}$$

Hence:

$$\left[\mathbf{S} \cdot \left[\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + y \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \right] \right]_G = G_1. \tag{5.111}$$

Thus:

$$y = \frac{-s_{23} + G_1}{s_{21}} \approx 0.25. \tag{5.112}$$

The value of y belongs to the interval [0, 1], hence using it one can find the point, which is sought:

$$[\mathbf{C_R} + y \cdot \mathbf{Y}]_R \approx 1.89. \tag{5.113}$$

Thus the approximate values of the R and G coordinates of this point are as follows:

$$R \approx 1{,}89 \text{ and } G \approx -0.52 \tag{5.114}$$

This point is denoted on the fig. 5.7 as **Y2**.

The approximate values of the X1 vertex are as follows:

$$R \approx 1{,}88 \text{ and } G \approx -0.52 \tag{5.114}$$

Its location is also shown on the fig. 5.7.

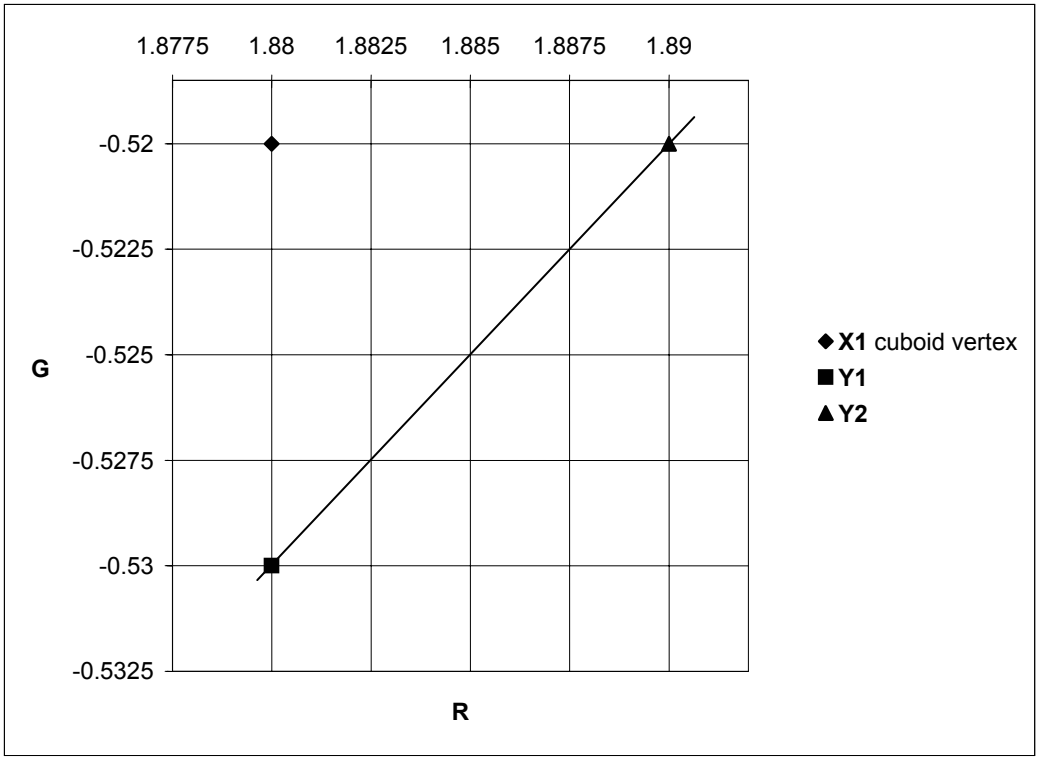

Fig 5.7. The analysis of the X1 cuboid vertex location relative the edge of the YCbCr cell.

When analyzing the fig. 5.7 one might think that the vertex **X1** is located outside the second YC_BC_R elementary cell, concluding that the first YC_BC_R elementary cell clipping cuboid and the second YC_BC_R elementary cell have no common part. However, this is not the case. The reasoning is conducted with the usage of the two decimal digits precision. Thus the situation presented on the fig. 5.8 is also possible.

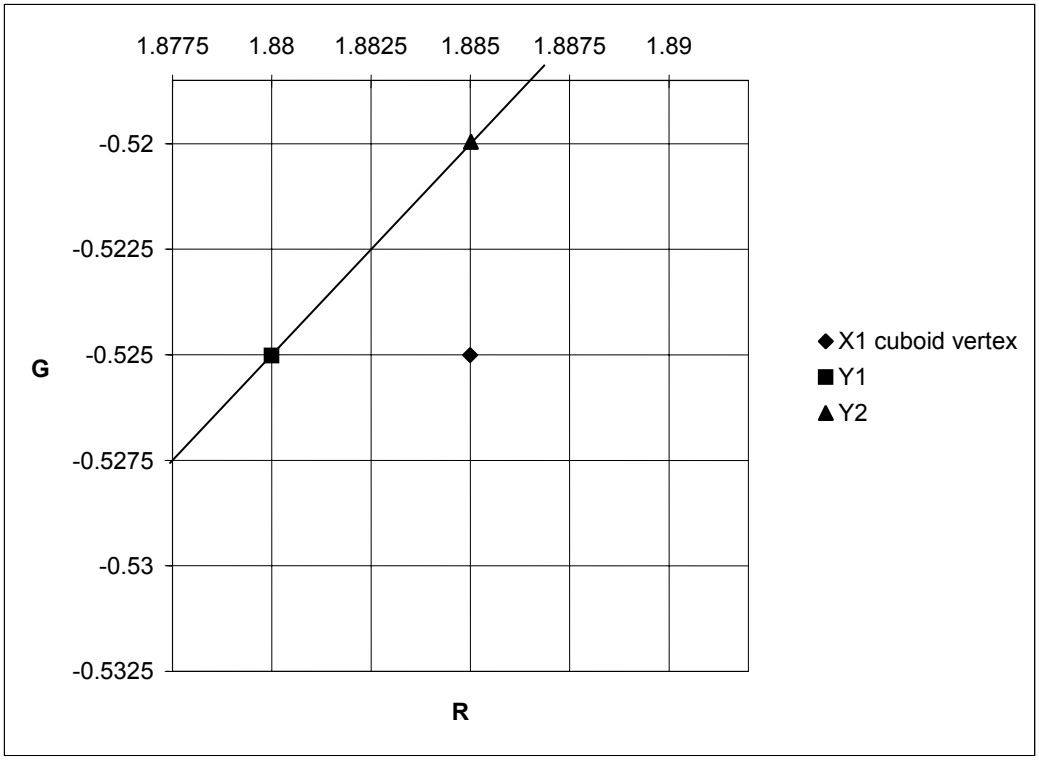

Fig 5.8. The analysis of the X1 cuboid vertex location relative the edge of the YCbCr cell.

The errors originating while floating point calculations can affect the result. The worst case must be taken into account, thus one must assume that the first YC_BC_R elementary cell clipping cuboid and the second YC_BC_R elementary cell have a common part. This part however is very small. Its dimensions are 0.01×0.01. On the fig. 5.6 such a tiny square is invisible. Hence the squares denoted by **a**, **b**, **c**, **d** and **X2** are much greater then necessary. The reasoning, similar to those presented in the chapter 5.1.1, must be conducted here. Provided the integer *RGB* point belongs to the area **X1**, no integer *RGB* point belongs to the area **X2**. This is why the color point can migrate along C_R axis for one cycle only. It may still however, migrate to another adjacent YC_BC_R elementary cell. All those, to which it can migrate, are so close to the B_{max} side face of the *RGB* range cube that no further range clipping can occur. of Hence it is proved that range clipping over the B_{max} side face of the *RGB* range cube may cause error accumulation up to the third encoding/decoding cycle. The reasoning for the B_{min} face of the *RGB* range cube is identical. The only difference is that the signs are opposite.

5.3.2 Edges of RGB range cube

The reasoning for the edges of the RGB range cube must be done in a similar way. The shape of the clipping cuboid for different kind of hyper-edges is of course different as shown in Fig. 5.9 for a corner of a two-dimensional X range cuboid.

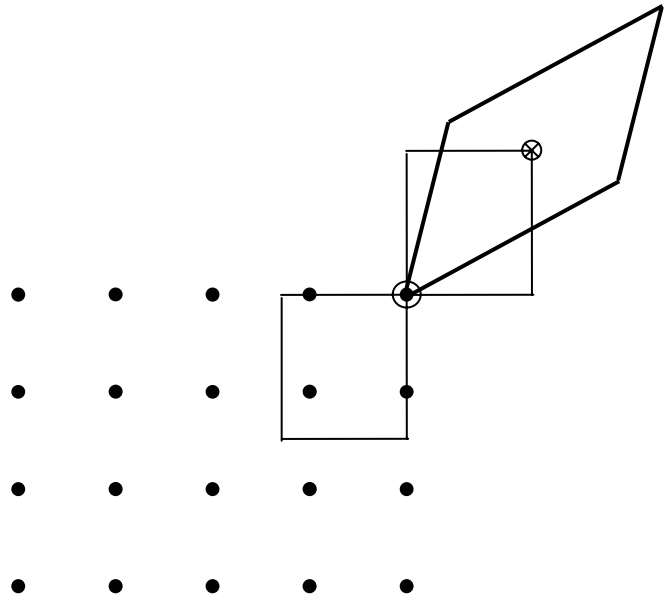

Fig 5.9. Two dimensional case - clipping rectangle (thin lines) for corner (empty circle) of X range cuboid (dots).

Two clipping cuboids for edges of a three-dimensional X (RGB) range cuboid are shown in Fig. 5.10. In order to analyze the greatest clipping errors, which occur along the edge of RGB range cuboid during YC_BC_R color transformation, it is necessary to consider eight YC_BC_R elementary cells, which form the parallelepiped, as shown in Fig. 5.11, with the center of this parallelepiped attached to the considered RGB range cuboid edge. Then, for every YC_BC_R elementary cell, belonging to the considered parallelepiped, the appropriate clipping cuboid must be constructed. Then it is necessary to find all the adjacent YC_BC_R elementary cells to which the data point can migrate from the analyzed YC_BC_R elementary cell. The above described procedure must be done for all the basic directions in RGB color space (direction R, direction G, direction B). After it is done for all the cells belonging to the considered parallelepipeds, it is found that no

range clipping can make the data point to leave these parallelepipeds (one of them is shown in Fig. 5.11).

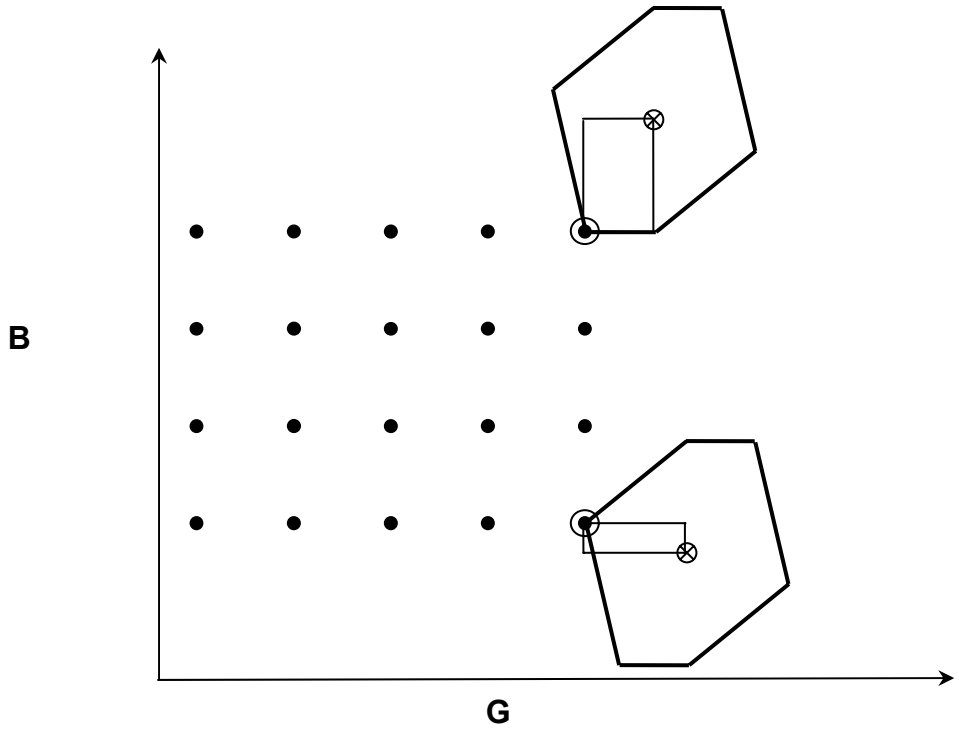

Fig 5.10. Cross-section of the RGB range cuboid (dots), YC_BC_R elementary cells (thick lines), projections and clipping cuboids projections(thin lines) drawn near edges (circles with dots).

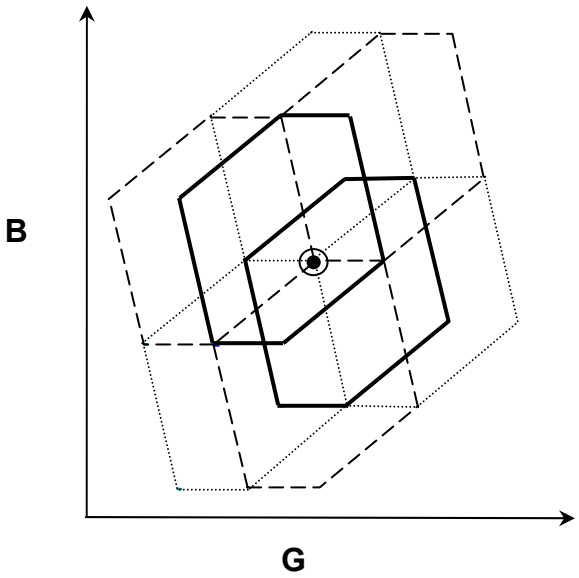

Fig 5.11. Projection of eight YC_BC_R elementary cells constituting a parallelepiped.

It is easy to prove that clipping errors, which occur along the edge of RGB range cuboid, cannot make the data point return to that YC_BC_R elementary cells, from which it has migrated away. Such a proof is given in the next paragraph. As the edge of RGB range cuboid, which crosses the considered parallelepiped, can visit at most four YC_BC_R elementary cells, the accumulation along such an edge can last for at most four cycles, which is an important result.

Let us prove that clipping errors, which occur along the edge of RGB range cuboid, cannot make the data point return to that YC_BC_R elementary cells, from which it has migrated away. Let us consider the range clipping which occurs along the RGB range cuboid edge, which is parallel to the R axis, and analyze Fig 5.12.

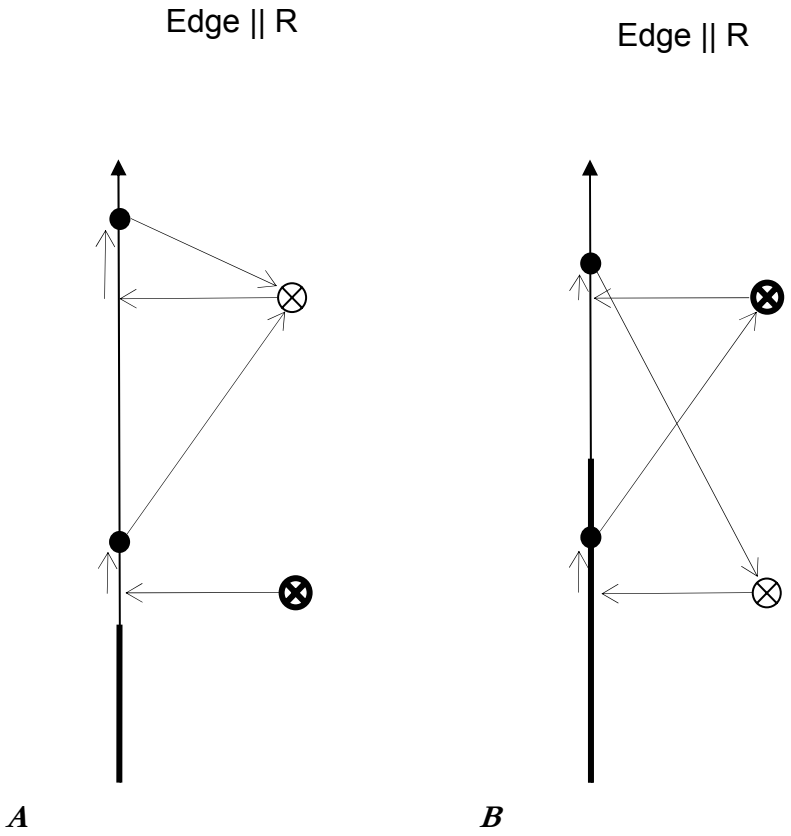

Fig 5.12. Range clipping along the edge, which is parallel to the R *axis.*

The RGB range cuboid edge is represented by a vertical arrow, which is drawn with a thick line at the bottom, and with a thin line at the top. The thick part of this edge belongs to that YC_BC_R elementary cell which has its center marked with a thick crossed circle. Similarly, the thin part of the edge belongs to that YC_BC_R elementary cell which has its center marked with a thin crossed

circle. Black dots denote RGB elementary cell centers. The arrows, which are drawn with faint lines, show the evolution of the data point. In Fig.5.12.A the data point migrates from the thick empty crossed circle, to the thin crossed circle, and cannot return. In Fig.5.12.B the data point migrates from the thick empty crossed circle, to the thin crossed circle and back. Hence the data evolution graph contains a "circuit". Note the substantial difference between cases A and B. In Fig. 5.12.A the thin crossed circle is above the thick crossed circle and the thin part of the RGB range cuboid edge is above the thick part of this edge. In Fig. 5.12.B the thick crossed circle is above the thin crossed circle and the thin part of the RGB range cuboid edge is above the thick part of the edge, hence the circles are interchanged.

Accordingly, in order to prove, that "circuits" do not occur during clipping the data alongside the edge of RGB range cuboid, it is enough to show that the case presented in Fig.5.12.B is impossible. In order to prove this, it is necessary to consider three following situations:

1. two adjacent YC_BC_R elementary cells separated by one of their side faces, Fig. 5.13,
2. two adjacent YC_BC_R elementary cells separated by one of their edges, Fig. 5.14,
3. two adjacent YC_BC_R elementary cells separated by one of their corners, Fig 5.15.

In Fig. 5.13 the RGB range cuboid edge which is parallel to the axis R and directed upwards is marked with a circle containing a black dot. The parallelogram in there is a projection of the side face separating the two adjacent YC_BC_R elementary cells from each other. The centers of these cells are marked with crossed circles. The two vectors spanning the parallelogram are the YC_BC_R base vectors. The vertical vector shows the location of the middle point of the parallelogram relative its specified corner (vertex). The third YC_BC_R base vector, and the opposite one are attached to the center of this parallelogram. They end in the centers of the considered adjacent YC_BC_R elementary cells. The two arrows, which are drawn with faint lines, show the clipping errors. Fig. 5.13 proves that for the two adjacent YC_BC_R elementary cells separated by one of their side faces, the situation shown in Fig. 5.12.B is impossible.

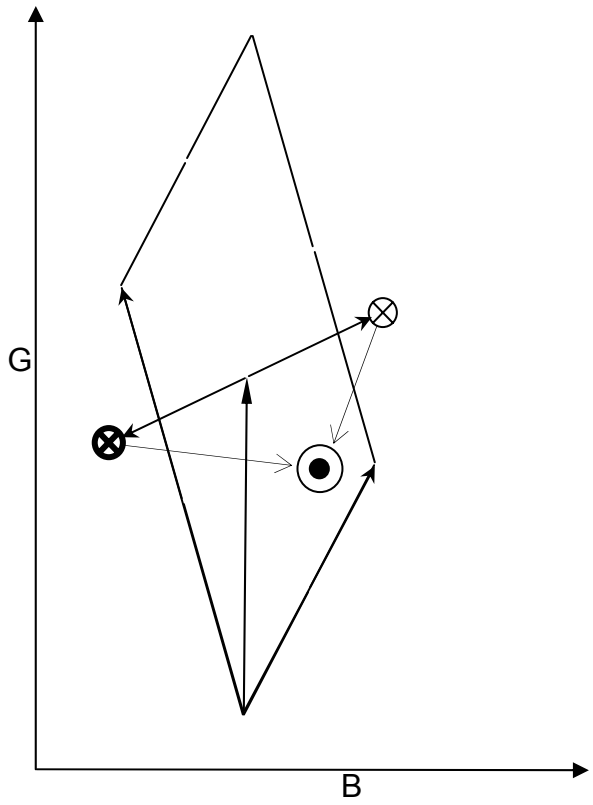

Fig 5.13. Side face separating two adjacent YC_BC_R elementary cells.

Also in Figures 5.14 and 5.15 the RGB range cuboid edge which is parallel to the axis R and directed upwards is marked with a circle containing a black dot and the two arrows, which are drawn with faint lines, show the clipping errors.

In Figure 5.14 the vertical line represent the edge separating the two adjacent YC_BC_R elementary cells and the vertical vector shows the location of its middle point relative its specified corner (vertex). The two slanted vectors, which are opposite to each other, show the locations of the adjacent YC_BC_R elementary cell centers relative the mentioned above middle point. Figure 5.15 follows the same convention. In the this figure the RGB range cuboid edge is passing through the corner which separates the adjacent YC_BC_R elementary cells.

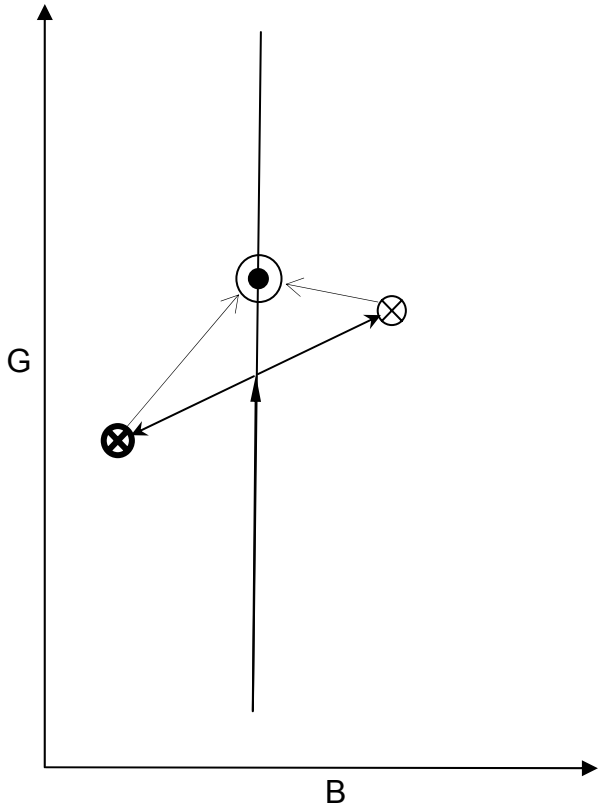

Fig 5.14. Edge separating two adjacent YC_BC_R elementary cells.

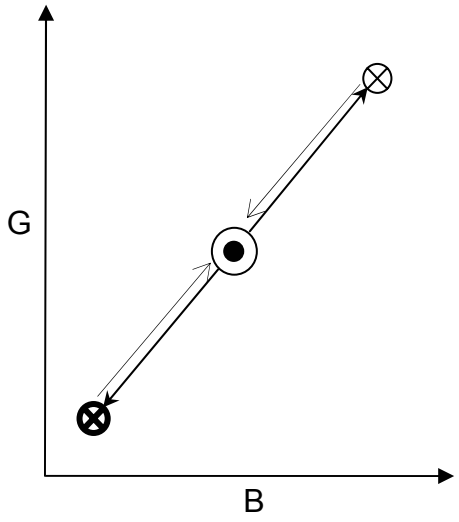

Fig 5.15. Corner separating two adjacent YC_BC_R elementary cells.

Figures 5.13, 5.14 and 5.15 prove that the situation shown in Fig. 5.12.B is impossible, hence there are no circuits in the data evolution graph for range clipping alongside of RGB range cuboid edges.

5.4 Conclusions

The Chapter 5 proves that as long the side faces of RGB range cube is considered the accumulation must end in the third encoding decoding cycle. The analysis of RGB range cuboid edges results in conclusion that the error accumulation can last for four cycles. Unfortunately it is not proved that a data point cannot migrate from a RGB range cuboid edge to its side face and back. If this were possible then the error accumulation would last form much longer. The analysis presented in this chapter does not depend on the dynamic range of the input data.

VI. LOSSY JPEG – ERROR ACCUMULATION WITH AND WITHOUT COLOR TRANSFORMATION

As described in Section 2.4 the block-wise discrete cosine transform (DCT) is used in JPEG. The block size is 8×8 pixels. The two dimensional DCT is separable so it can be done with one dimensional DCT, which is represented by 8×8 matrix. Such transform is first applied to rows and then to columns (or vice versa). The picture elements can be rearranged to create a one dimensional sample vector. As each picture element in a block is distinguished with two indices, it is necessary to map those pairs of indices into a single index.

Hence a frequency sample F_{ij} indexed by i and j, gets index k in the following way:

$$F_k = F_{i\times 8+j}, \tag{6.1}$$

provided $i, j \in \{0,1,...,7\}$.

Mapping index k into the pair of indexes i and j is equally simple:

$$F_{i,j} = F_{k \bmod 8, \lfloor k/8 \rfloor} \tag{6.2}$$

where "mod" stands for "modulo".

The luminance samples must be rearranged in the same way. Both these rearrangements imply the necessity of finding the appropriate elements for the 64×64 DCT matrix. Such forward and backward transform matrices (without quantization) have the following values of infinity norms and determinants:

$$L_\infty(\mathbf{DCT}) = 8.0 \tag{6.3}$$

$$L_\infty(\mathbf{DCT}^{-1}) = 6.98 \tag{6.4}$$

$$|\det(\mathbf{DCT})| = |\det(\mathbf{DCT}^{-1})| = 1 \tag{6.5}$$

Equation **(6.4)** shows that DCT may not be reversible (see Section 4.4, Equation 4.49 – sufficient condition for reversibility is not satisfied). However, equation **(6.5)** shows that DCT may be reversible (see Proposition 1, Section 4.3 – necessary condition for reversibility is satisfied). However, the experiments show it is not. Moreover, as shown in [Hao1], the equation **(6.5)** tells

that DCT can be factorized in such a way, that it will surely be reversible. As both the forward transform and the inverse one have the infinity norm greater then one, it is very improbable that the error saturation will be achieved in few encoding/decoding cycles.

If the frequency samples are quantized, then:

$$L_\infty(\mathbf{qDCT}) < 0.74, \tag{6.6}$$

$$L_\infty(\mathbf{qDCT}^{-1}) > 444.74, \tag{6.7}$$

where:

$$\mathbf{qDCT} = \frac{1}{f} \cdot \mathbf{Q}^{-1} \cdot \mathbf{DCT}, \tag{6.8}$$

$$\mathbf{qDCT}^{-1} = f \cdot \mathbf{DCT}^{-1} \cdot \mathbf{Q}. \tag{6.9}$$

The matrix **Q** is the luminance quantization table defined in an informative part of JPEG norm, which is appropriately rearranged into 64×64 diagonal matrix. Coefficient f is a scale factor, which is greater then one. The values shown in **(6.6)** and **(6.7)** were found for $f = 1$. The inequality **(6.7)** shows that such a transform may be lossy, what is obvious. Moreover:

$$\left|\det(\mathbf{qDCT})\right| = \frac{1}{f} \cdot \left|\det(\mathbf{Q}^{-1})\right| \cdot \left|\det(\mathbf{DCT})\right| = \frac{1}{f} \cdot \left|\det(\mathbf{Q}^{-1})\right| << 1, \tag{6.10}$$

which means that such a transform is lossy (see Proposition 1, Section 4.3 – necessary condition for reversibility is not satisfied). The inequality **(6.6)** tells, that if range clipping is forbidden in the forward DCT transform then in n-th encoding/decoding cycle the error saturation will be achieved, provided the range clipping in the inverse transform does not occur in that cycle (see Section 4.5, Proposition 3). In order to see, if the range clipping is likely to happen during the inverse transform, let us compare the size of the luminance range cube (X range cuboid) with the lengths of the quantized DCT elementary vectors (Y elementary vectors). One can find such lengths summing up all the squared elements for each column of $\mathbf{qDCT}^{-1}$ matrix and taking its square root. Each column gives the length of the one elementary DCT vector. Their lengths vary

between 10 and 121, which is implied be the luminance quantization matrix. If the luminance samples are stored in the variables with 8-bit precision, then each such sample cannot be greater then 255. Hence all the edges of the luminance range cuboid have lengths 255. As the length of the luminance range cube edge (255), and the length of longest DCT base vector (121) have the same order of magnitude, one can expect the phenomena occurring on the luminance range cube boundaries will be significant. Hence the substantial range clipping will take place for many cycles. The theoretical analysis of the error accumulation in multiple encoding/decoding cycles for DCT is much more difficult then for the color transformation, because it necessary to consider the 64-dimensional X range cuboid. This is why only experimental analysis for the error accumulation in DCT multiple encoding/decoding cycles was done.

Fig. 6.1 presents results for multiple encoding and decoding of the image Tiffany with JPEG technique. These results are very typical and the other test images behave in the same way. Image quality loss was measured while multiple encoding and decoding with coder settings kept constant. Every image component (red, green and blue) was processed independently and the PSNR values shown on Figure 6.1 are those averaged over all the three components.

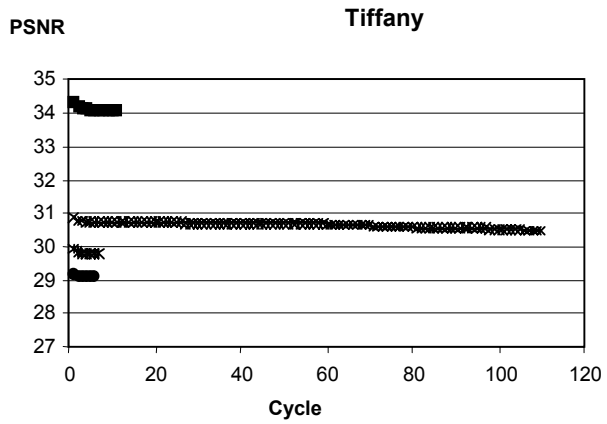

Fig 6.1. Quality decrease for repeatedly JPEG-compressed and decompressed image Tiffany

Error saturation has been achieved in each explored case but, as mentioned above, it may take many encoding/decoding cycles until the error accumulation ends. The highest quality series on Fig. 6.1 are obtained for transformation without quantization. Usually the better quality is demanded more cycles are necessary to achieve errors saturation. There are some exceptions to this rule however what can be seen on the Figure 6.1.

When compressing the color images with JPEG, the DCT is usually preceded with color transformation. Standard YC_RC_B color transformation is in common use. For the simplified case of 2×2 block of data: $\begin{bmatrix} R_1 & R_2 \\ R_3 & R_4 \end{bmatrix}, \begin{bmatrix} G_1 & G_2 \\ G_3 & G_4 \end{bmatrix}, \begin{bmatrix} B_1 & B_2 \\ B_3 & B_4 \end{bmatrix}$, (subscript denotes here an index of a pel in a block) such operations can be presented as follows:

$$\begin{bmatrix} Fy_1 \\ Fy_2 \\ Fy_3 \\ Fy_4 \\ Fcb_1 \\ Fcb_2 \\ Fcb_3 \\ Fcb_4 \\ Fcr_1 \\ Fcr_2 \\ Fcr_3 \\ Fcr_4 \end{bmatrix} = \text{round}\left(\mathbf{3DCT} \cdot \mathbf{P} \cdot \text{round}\left(\mathbf{4T} \cdot \begin{bmatrix} R_1 \\ G_1 \\ B_1 \\ R_2 \\ G_2 \\ B_2 \\ R_3 \\ G_3 \\ B_3 \\ R_4 \\ G_4 \\ B_4 \end{bmatrix} \right) \right), \qquad \textbf{(6.10)}$$

where: $\mathbf{3DCT} = \begin{bmatrix} \mathbf{DCT} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{DCT} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{DCT} \end{bmatrix}$, $\mathbf{O} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, $\mathbf{4T} = \begin{bmatrix} \mathbf{T} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{T} \end{bmatrix}$,

$\mathbf{0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, $\mathbf{T}$ is the YC_RC_B color transformation matrix, **DCT** is a 4×4 matrix for the 2×2-point discrete cosine transform in two dimensions (the used 2D-DCT is usually separable so

it can be done by means of 1D-DCT), **P** is a permutation matrix and the left hand side vector in Equation **(6.10)** consists of frequency samples for luminance and chrominance. In JPEG the 8×8 blocks of data are used, hence all the matrices should be appropriately changed, which means that instead of using **4T** and 2×2-point **DCT**, the **64T** and 8×8-point **DCT** would have to be used. If there were no rounding operation after the color transformation in the Equation **(6.10)** then this case would be only slightly more complex then the *DCT* without any color transformation. Then the conditions, given in the proposition 1, 2 and 3, would have to be checked for the matrix product $\mathbf{3DCT} \cdot \mathbf{P} \cdot \mathbf{64T}$.

As there are two rounding operations in Equation **(6.10)**, the theoretical analysis is much more difficult. Hence the problem was examined experimentally. Figure 6.2 presents image quality decrease for exemplary test image "Lena" while multiple JPEG encoding and decoding combined with color transformation. Transformed image components are subsequently compressed with JPEG coder and all the operations are then reversed.

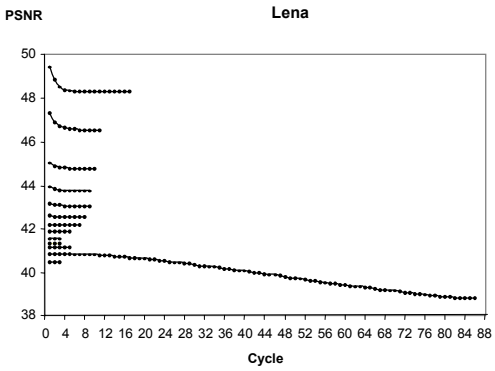

Fig 6.2. Multiple encoding and decoding of "Lena" image – JPEG compression combined with color transformation (behavior of the other test images is similar).

If color transformation is combined with JPEG compression, then for majority of cases the error saturation is also observed after a limited number of encoding/decoding cycles. However, for some coder settings an interesting new phenomenon is observed. After some

number of encoding/decoding cycles the sequence of images is received in which the first and the last image are the same. Hence a new kind of saturation is achieved. If repetitive encoding and decoding were done then the image would always follow the same path coming back to the same point, hence no further errors are introduced into the image. This phenomenon is presented on the Figure 6.3.

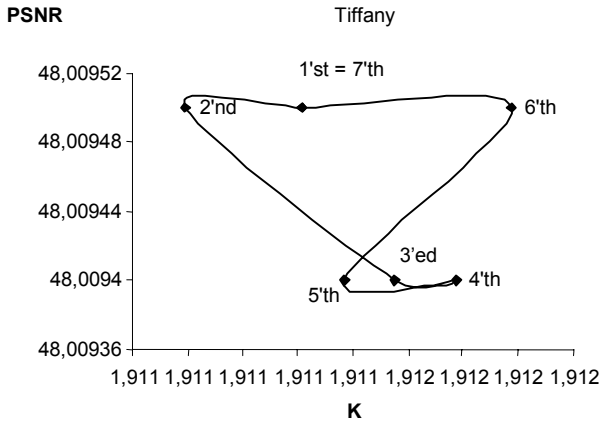

Fig 6.3. Image Tiffany – cyclic sequence of images, K – compression ratio

When checking thoroughly which part of the image Tiffany is responsible for such a behavior, it was found that a single 8×8 block of pels evolves in such a way. During experiments on other natural images it was confirmed that this phenomenon is very rare, and happens for few blocks of a natural image only. Using the graph theory terminology such a sequence of images can be called "circuit".

The results presented in this chapter were published in [Rakowski].

VII. NEAR LOSSLESS TECHNIQUES BASED ON PREDICTIVE COMPRESSION

7.1 Lossy compression based on linear prediction

Intra-picture linear prediction can be treated as a special case of linear transformations. This is because one can construct a matrix which, after being multiplied by an input signal vector, gives the prediction error vector. Lossless image compression widely exploits linear prediction (Section 2.2). Similarly, lossy compression may be also implemented as intra-picture linear prediction followed by quantization and binary coding. Quantizing a prediction error vector is equivalent to multiplying it by a diagonal quantization matrix and rounding the results. If the prediction is done on the signal consisting of N samples, then the N×N matrix is to be constructed. For casual predictors this matrix is lower triangular matrix. Of course the signal may be partitioned into smaller blocks in which the prediction is done. This is exactly the same situation as for the transform coding. All the theorems hitherto presented can be used to analyze it. The JPEG-LL, which is a lossless compression standard, uses linear predictors. It is easy to introduce a nearly lossless mode of operation into this technique by quantizing the prediction error. The JPEG-LL, modified in such a way, could be theoretically analyzed, as it was presented in previous chapters.

7.2 Near lossless JPEG-LS error accumulation

The JPEG-LS (LOCO) is briefly described in Section 2.3, in which lossless image compression standards are presented. The nearly lossless mode of operation is defined in this standard, what is achieved by the prediction error quantization. Such a technique makes it possible to control the maximum difference between every original image sample and the corresponding one in the reconstructed image. Among LOCO initial predictors only one is linear. But as this initial prediction is further modified, the prediction used in this algorithm is not linear at all. This is why

the analytical methods, which were developed in previous chapters, cannot be used to analyze this algorithm.

Error accumulation for the JPEG-LS nearly-lossless mode of operation for image Lena in *RGB* color space (without any color transformation) is shown on Figure 7.1. The top curve is for maximum error d_{max} set to one, and the lower one is for maximum error d_{max} set to two. Every point on the plot is averaged over three (red, green, blue) components.

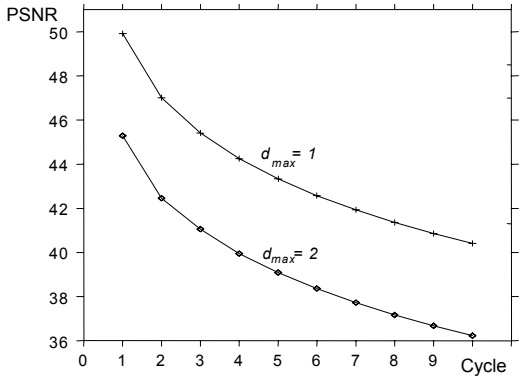

Fig 7.1. Near lossless LOCO for image Lena – PSNR – error accumulation for maximum error equal one and two respectively (the other explored images behave similarly) – RGB color space

This plot proves that the accumulated errors are significant. Already the second cycle reduces the quality about 2-3 dB as compared to the first cycle. Similar plots have been obtained for other test images. The loss of 10 dB of the peak signal-to-noise ratio after about 10 cycles of encoding and decoding is common. The other explored images behave similarly, as it can be seen in the Table 7.4 in the next section, where more precise data for "Lena" and "Clown" test images are presented. In the near lossless LOCO encoder the error accumulation is substantial and no error saturation is achieved. Maximum error for few encoding/decoding cycles never exceeds maximum error of a single encoding decoding cycle multiplied by the number o cycles, provided that no color transforms are performed.

Error accumulation for the JPEG-LS nearly lossless mode of operation with d = 1 and 2 for image Lena in YC_BC_R color space is shown in Figures 7.2 and 7.3. The values of PSNR [dB] and maximum accumulated error are plotted versus cycle number. The results for other test images are similar and they prove the conclusions are right .

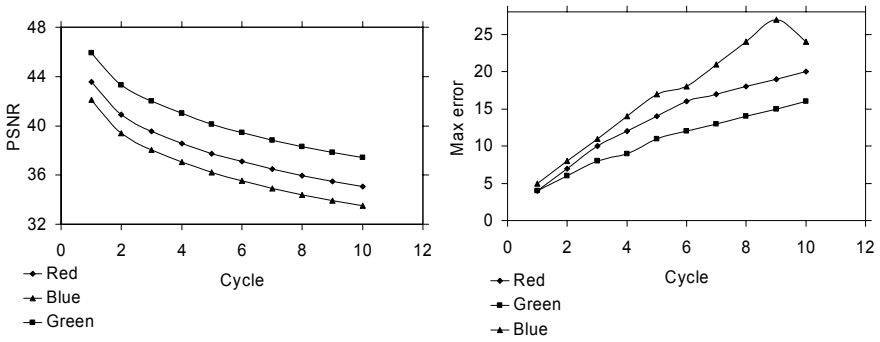

Fig 7.2. Near-lossless JPEG-LS with d =1 in YC_BC_R color space: PSNR [dB] and maximum accumulated error for test image "Lena".

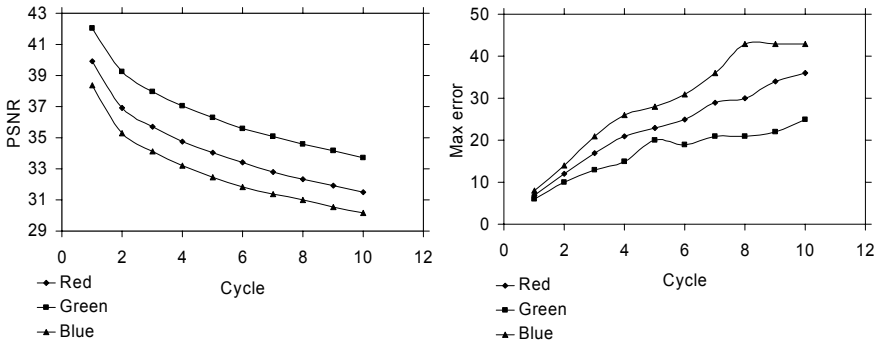

Fig 7.3. Near-lossless JPEG-L with d =2 in YC_BC_R color space: PSNR [dB] and maximum accumulated error for test image "Lena".

The plots show that, unlike before, the maximum error for few encoding/decoding cycles is not bounded by the product of a single encoding/decoding cycle error and a number of cycles. This is caused by the interference of color transform rounding errors with the errors due to prediction quantization. The PSNR plots show the quality monotonically decreases as the consecutive encoding/decoding cycles are done.

7.3 New method of nearlossless still image compression with no error accumulation

Quantizing YC_RC_B color components combined with lossless coder such as JPEG-LS or CALIC is proposed as a means of a rate-distortion control. If the color transformation properly de-correlates the color components then compression efficiency will be improved. Tables 7.1 and 7.2 show the experimental results which prove that color transformations are suitable for that purpose. Table 7.3 shows the quality loss due to rounding in color YC_RC_B transformation.

TABLE 7.1. COMPRSSION RATIO FOR NARUAL IMAGES

Image	JPEG-LS	CALIC	JPEG-LS	CALIC
	RGB 8 bit representation		JPEG-2000 reversible 8/9 bit representation	
Airplane	2.02	2.06	2.31	2.36
Baboon	1.30	1.33	1.45	1.49
Boats	1.53	1.57	1.73	1.77
Clown	1.68	1.73	1.84	1.88
Lena	2.26	2.33	2.47	2.55
Penguin	1.50	1.62	1.87	2.01
Peppers	1.69	1.73	1.79	1.84
Sailboat	1.53	1.56	1.65	1.69
Average	1.64	1.70	1.89	1.94
	YC_BC_R 8 bit representation		YC_BC_R 10 bit representation	
Airplane	2.74	2.76	2.10	2.05
Baboon	1.56	1.60	1.42	1.40
Boats	1.96	2.01	1.67	1.64
Clown	2.11	2.16	1.77	1.73
Lena	2.98	3.04	2.23	2.17
Penguin	2.04	2.17	1.87	1.75
Peppers	2.01	2.06	1.71	1.67
Sailboat	1.84	1.89	1.61	1.58
Average	2.08	2.13	1.76	1.72

Table 7.2. Compression ratio for histological images

Image	JPEG-LS	CALIC
	RGB 8 bit representation	
c40_1	2.46	2.63
Desm1	2.52	2.63
desm15	2.60	2.68
Desmvg1	2.69	2.78
Desmvg6	2.51	2.58
e25_1	2.84	3.01
sarco1	1.94	2.02
sarco8	2.10	2.18
	YC_BC_R 8 bit representation	
c40_1	3.29	3.41
desm1	3.88	3.96
desm15	3.98	4.04
Desmvg1	4.17	4.25
Desmvg6	3.76	3.77
e25_1	3.85	3.92
sarco1	3.03	3.08
sarco8	3.02	3.08

TABLE 7.3. PSNR FOR THE RGB → YC_RC_B →RGB TRANSFORMATION WITH 8-BIT SAMPLES FOR BOTH NATURAL AND HISTOLOGICAL IMAGES

Image	Red	Green	Blue
Airplane	52.0	53.9	51.0
Baboon	51.9	53.9	51.0
Boats	52.0	54.0	51.1
Clown	52.1	54.0	51.2
Lena	51.9	53.9	51.0
Penguin	52.0	53.2	52.3
Peppers	51.9	54.0	51.1
Sailboat	51.9	53.9	50.1
Histological images (average)	52.0	53.7	51.0

The proposed method has proved to be successful as long as data fidelity is considered important since error saturation is achieved. Quantizing YC_RC_B color components introduces distortions into the image in first two encoding/decoding cycles. The coarser color components quantization is applied, the better compression ratio is achieved in the next stage of image processing. Lossless compression does not introduce any further distortions into the image.

Near-lossless mode of JPEG-LS in the *RGB* and YC_BC_R color spaces is compared with lossless compression in the YC_BC_R space with reduced precision of samples (Fig. 7.4 and 7.5, Table 7.4 and 7.5).

TABLE 7.4. NATURAL IMAGES

Coder	**Color Space**	**Test image *Lena***								**Test image *Clown***							
		K	***d_{max}***			**PSNR [dB]**				***K***	***d_{max}***			**PSNR [dB]**			
			R	**G**	**B**	**R**	**G**	**B**	**Av**		**R**	**G**	**B**	**R**	**G**	**B**	**Av**
Lossless	RGB	2.26	0	0	0	∞	∞	∞	∞	1.68	0	0	0	∞	∞	∞	∞
Lossless	JPEG 2000 reversible	2.47	0	0	0	∞	∞	∞	∞	1.84	0	0	0	∞	∞	∞	∞
Lossless	10 bits luminance 10 bits chrominance YC_BC_R	2.23	0	0	0	∞	∞	∞	∞	1.77	0	0	0	∞	∞	∞	∞
Lossless	8 bits luminance 8 bits chrominance YC_BC_R	2.98	1	1	2	51.9	53.9	51.0	52.3	2.11	1	1	2	52.1	54.0	51.2	52.4
Lossless	8 bits luminance 7 bits chrominance YC_BC_R	3.43	2	2	3	47.9	51.4	46.3	48.5	2.33	2	2	3	47.7	51.5	46.5	48.6
Lossless	7 bits luminance 7 bits chrominance YC_BC_R	3.81	3	2	2	46.7	49.0	45.3	47.0	2.48	3	2	3	46.6	49.1	45.6	47.1
Lossless	6 bits luminance 6 bits chrominance YC_BC_R	4.83	5	5	6	40.9	43.4	39.6	41.3	3.05	6	5	6	40.8	43.3	40.1	41.4
Near-lossless $d_{max}=1$ 1st cycle	RGB	3.79	1	1	1	50.0	50.0	50.0	50.0	2.46	1	1	1	49.9	49.9	50.0	49.9
Near-lossless $d_{max}=1$ 10th cycle	RGB	3.13	10	10	10	40.5	40.4	40.3	40.4	2.35	10	10	10	40.3	40.3	40.4	40.3
Near-lossless $d_{max}=2$ 1st cycle	RGB	5.01	2	2	2	45.5	45.2	45.2	45.3	3.10	2	2	2	45.1	45.2	45.2	45.2
Near-lossless $d_{max}=2$ 10th cycle	RGB	3.82	17	18	18	36.5	36.2	36.0	36.2	2.77	17	18	17	35.6	35.6	35.8	35.7

TABLE 7.5. HISTOLOGICAL IMAGES

Technique	Test image *Sarco1*							Test image *Desm1*						
	K	δ_{max}			PSNR[dB]			K	δ_{max}			PSNR[dB]		
		R	G	B	R	G	B		R	G	B	R	G	B
Lossless with color transformation	3.03	1	1	2	51.8	54.0	51.01	3.88	1	1	2	52.0	54.0	50.9
Nearly lossless with $\delta_{max} = 1$	3.10	1	1	1	49.9	49.8	49.8	4.71	1	1	1	49.9	50.0	49.9
with $\delta_{max} = 2$	4.01	2	2	2	45.1	45.1	45.1	6.47	2	2	2	45.6	45.5	45.4
with $\delta_{max} = 3$	4.73	3	3	3	42.2	42.1	42.1	8.09	3	3	3	42.9	42.8	42.7

For the near-lossless JPEG-LS a significant decrease of compression ratio K is observed (see Fig. 7.5), which is due to coding of accumulated errors. These errors are very noise-like so it is difficult to compress them. The application of near-lossless JPEG-LS in YC_BC_R color space results in *K* up to 60 % greater than in the *RGB* color space. The image quality is then respectively worse, as it can be seen on the Fig. 7.4. For example, after the second encoding/decoding cycle the near lossless JPEG-LS in the *RGB* color space with d=1 is inferior to lossless JPEG-LS with 7-bit samples of Y, C_B, C_R. Here, "inferior" means lower compression ratio and lower quality of the reconstructed image (both Fig. 7.4 and 7.5 must be used to see it). In general, similar conclusions are valid for $d = 2$ and for the YC_BC_R color space and for corresponding number of bits of YC_BC_R representation used for lossless compression. Note that PSNR lower than about 40 dB usually corresponds to visible image degradations. Such compression cannot be described as near-lossless. The degradation caused by rounding of YC_BC_R is related to appearance of false contours while near-lossless compression introduces more noise.

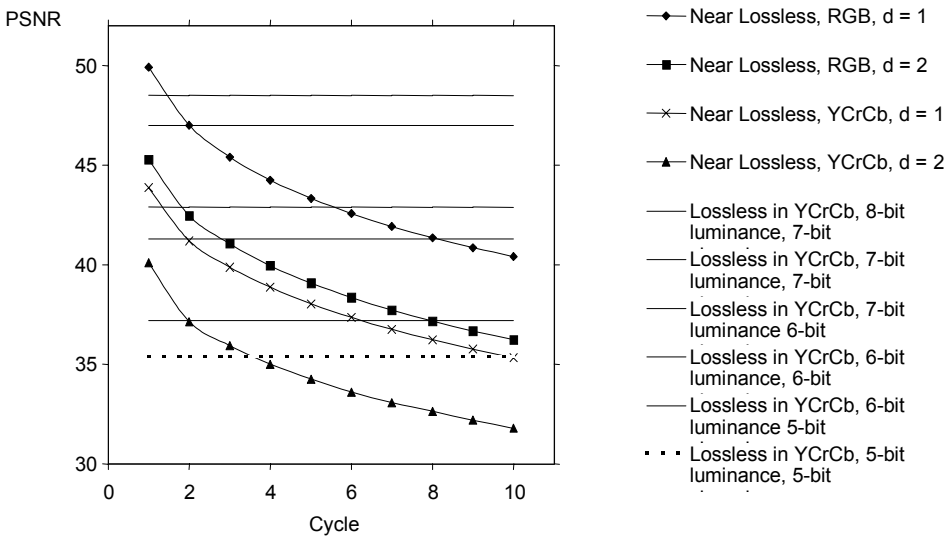

Fig 7.4. Error accumulation in multiple encoding-decoding cycles for LOCO coder combined with color transformation for image Lena (the other images behave in the same way).

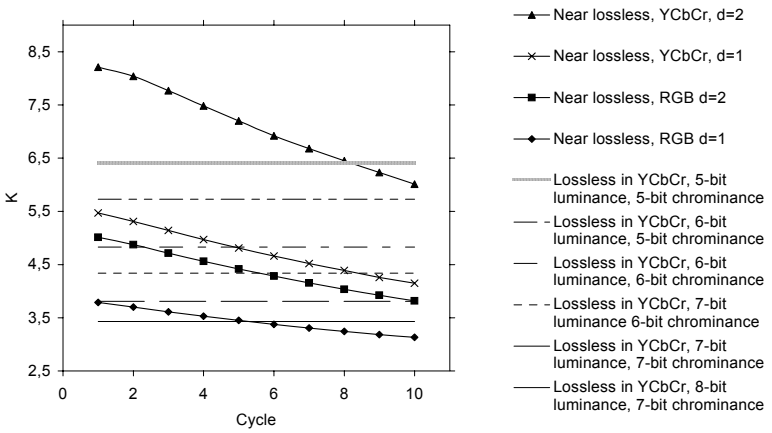

Fig 7.5. Compression ratio in multiple encoding-decoding cycles for LOCO coder combined with color transformation for image Lena (the other images behave in the same way).

The experiments prove that standard near-lossless JPEG-LS compression technique suffers from error accumulation in the consecutive cycles of coding and decoding. The errors accumulate after each cycle and large errors occur after only few cycles. Moreover the cumulated coding noise reduces compression ratio in the consecutive cycles.

The proposed technique is advantageous because the color transformation $RGB \rightarrow YC_RC_B$ introduces errors in the first cycle only. Further cycles do not produce rounding errors (the error accumulation for color transformation is carefully analyzed in Chapter 5). Therefore, for multiple compression cycles, lossless JPEG-LS with reduced sample precision in YC_RC_B is superior to near-lossless JPEG-LS.

The results presented in this chapter were published in [Domanski2], [Domanski3] and [Domanski4].

VIII. VIDEO CODING

There are many hybrid video coding standards. They all use three basic techniques: motion compensated prediction, transform coding, and entropy coding. The only stage which introduces errors into the data is transform coding. But as long as errors accumulation is concerned the motion compensated prediction is also crucial, since it makes the coding strongly non-linear. This is because any time the prediction is done, the position of an encoded data block, which is used to predict another data, relative actually encoded block can be changed. Besides, the criterion used for choosing the best prediction is usually MAD (Minimum Absolute Difference). This is why only experimental analysis of such coders is presented in this dissertation.

Error accumulation in two video coding standards H263 and MPEG-2, which are the most representative examples of hybrid video coding standards, was tested. The "Digital Video Coding group at Telenor R&D" and MPEG Software Simulation Group – Test Model 5 implementations were chosen. Results for CIF resolution sequence "cheer" are presented (the results for other sequences are similar). Sequence consists of 298 frames and is represented in YC_BC_R color space, hence no color transformation is done while consecutive encoding/decoding cycles. Of course it must be performed in order to play the sequence on the screen but the color transformation rounding errors do not influence the error accumulation. The chrominance decimation format is 4:2:0. Quantizer scale ranges form 2 to 62. Rate control was switched off during the experiments hence Q was constant while the sequence was being coded.

Figures 8.1 and 8.2 present the results for H.263 codec.

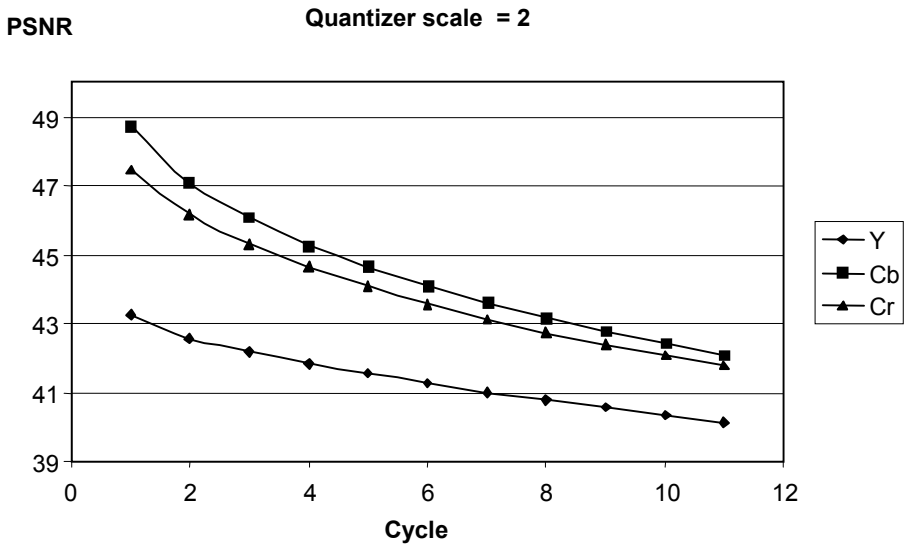

Fig 8.1. Error accumulation for "cheer" CIF-sequence H.263 codec, quantizer scale equal 2

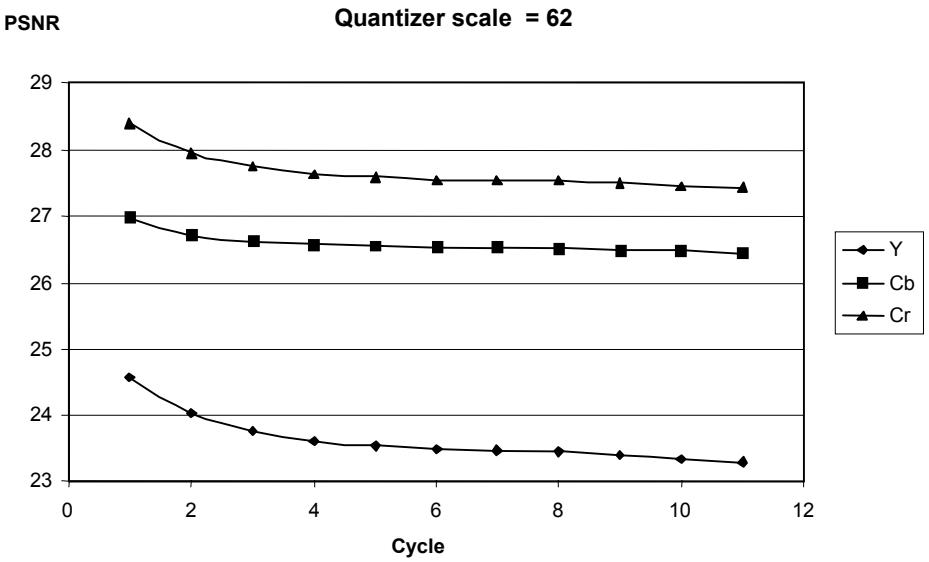

Fig 8.2. Error accumulation for "cheer" CIF-sequence H.263 codec, quantizer scale equal 62

For higher qualities (Fig. 8.1) the PSNR decreases rapidly for the first few encoding/decoding cycles. For each subsequent cycles the PSNR loss seams to decrease. The plot for lower qualities (Fig. 8.2) is much more flat, but also here the PSNR loss for the first few encoding/decoding cycles is greater then for subsequent ones. Error saturation was not observed. On both the above

figures the chrominance PSNR is better then luminance PSNE. This is commonly known phenomenon and it caused by smaller dynamic range of the chrominance components.

Figures 8.3 and 8.4 present the results of experiments with MPEG-2, which are analogues to those made for H.263.

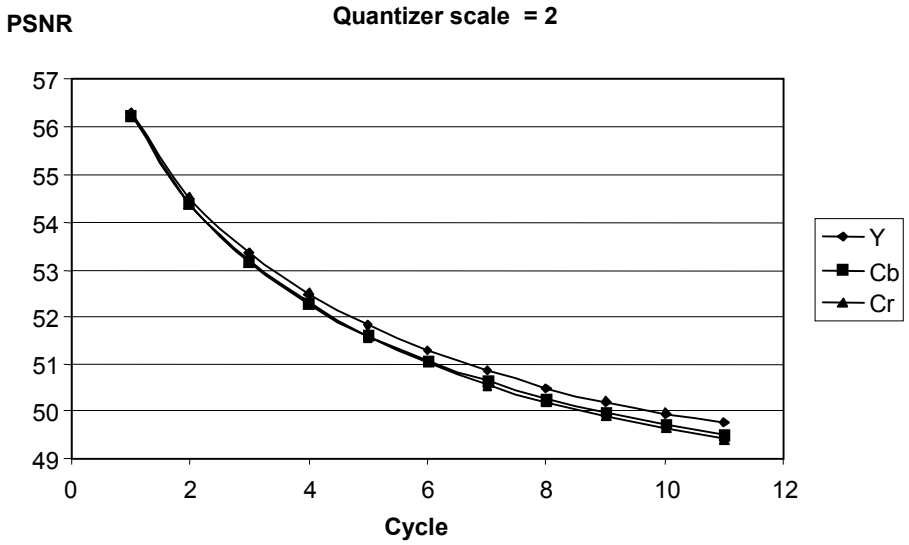

Fig 8.3. Error accumulation for "cheer" CIF-sequence MPEG-2 codec, quantizer scale equal 2

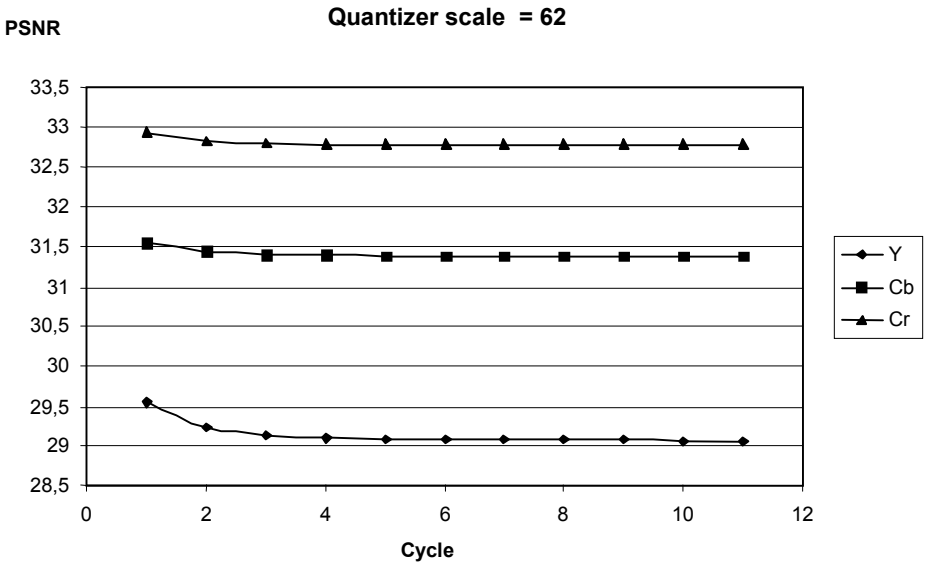

Fig 8.4. Error accumulation for "cheer" CIF-sequence MPEG-2 codec, quantizer scale equal 62

Also for MPEG-2 we can see faster quality decrease for higher qualities and lack of error saturation.

The experimental results for all the video sequences look similar. The main difference between the transform coding of still images and video sequences is the motion compensation used in video sequences coding. This is probably reason why neither error saturation nor circuits (closed cycles) were observed. If circuits exist then either they will occur after many cycles or their period will be very long.

IX. CONCLUSIONS

The dissertation presents a theoretical and experimental analysis of rounding error accumulation while multiple encoding/decoding cycles for linear transformations and other topics related to it. The problem of transformation reversibility is one of those; hence the necessary and the sufficient condition for the transformation reversibility are presented (see Sections 4.3 and 4.4). Of course these reversible ones do not cause error accumulation as they do not introduce any errors into the data.

The theorem giving the upper bounds for the rounding errors which can occur in a single encoding/decoding cycles is presented. It is easy to find these bounds for all the linear transformations (see Section 4.4), provided there is only one rounding while the encoding operation and one rounding during the decoding operation.

The condition for the error accumulation terminations is given (Section 4.5). Having this condition stated it is possible to analyze a specific linear transformation theoretically. The principles for such an analysis are worked out and presented in Chapter 4. The system of difference equations is derived and a way towards its solution is proposed. It is shown that a general solution, with all the specific solutions included in it, has a directed graph structure.

A detailed theoretical analysis of the YC_BC_R color transformation is done. The upper bounds for rounding errors are found theoretically and verified experimentally (Chapter 5). It is found that error accumulation for original components (R, G, B) with dynamic range [0, 255] is limited to the first two encoding/decoding cycles provided all the components (R, G, B, Y, C_B and C_R) are stored in 8'bit variables. It is proved by checking the error accumulation for all the possible RGB triples having the above defined dynamic range. Moreover, the experiments prove that the error accumulation in the second encoding/decoding cycle is negligible for natural images.

In Chapter 5 more sophisticated reasoning for the YC_BC_R color transformation is also presented. The conclusions drawn in it do not depend on the dynamic range of input data. It shows that error accumulation is not likely to last for more then the first four encoding/decoding cycles.

An error accumulation for DCT based still image compression standard JPEG is experimentally explored (Chapter 6). The results are very interesting. The two cases were researched – with and without a color transformation with constant quantization step size.

In the first case (with RGB→YC_BC_R color transformation), error accumulation ends after a finite number of encoding/decoding cycles for majority of natural test images. In some cases however, circuits were observed. Such a circuit consisted of a sequence of images, in which the first one and the last one were the same.

In the second case (without any color transformation), for all the explored test images, the error accumulation ended after a finite number of encoding/decoding cycles. It is not yet possible to determine the number of these cycles before the experiment is done. It depends on the input data and the coder settings. Usually the higher image quality demands are set in the encoder the more encoding/decoding cycles are necessary to achieve the error saturation. But there are some exceptions to this rule. No circuits (term taken from graph theory) are observed.

An error accumulation for the most representative case of near-lossless compression standard JPEG-LS (LOCO) was experimentally explored and proved to be substantial (Chapter 7). No error saturation was observed for this technique. The new technique for near-lossless compression is proposed, which exploits YC_BC_R color transformation characteristics. Achieving error saturation after the second encoding/decoding cycle is its main advantage.

The main original theoretical results of the dissertation are the following:

- Theorems on transformation reversibility and error accumulation (partially with the advisor) (Sections 4.3 – 4.5). In particular:
    - Necessary and sufficient conditions for transform reversibility are presented.
    - The upper bounds for rounding errors are derived.
    - The condition for the error accumulation termination is derived.
- Error accumulation analysis in multiple encoding/decoding cycles (Section 4.6) used for YC_BC_R color transformation analysis (Chapter 5).
- A proposal for a near-lossless image compression technique with no error accumulation (partially with the advisor) – Section 7.3.

Other important results have been obtained experimentally:

- Theoretical and experimental analysis of error accumulation in transform coding (JPEG)
- Experimental results which proved error accumulation in near-lossless JPEG-LS technique.

The following topics are still to be explored in the nearest future:

- Theoretical analysis of the observed elementary cycles in the JPEG compression alone and combined with color transform.
- The interconnections between trees of solutions for different side faces of the X range cuboid in linear transformation giving a graph representing a general solution.

Hence, the conditions for error accumulation in compression and color transformation have been formulated and used in image processing analysis. The cases which were too complex for theory were examined experimentally.

References

Articles

[Blume1] H. Blume, A. Fand, "Reversible and irreversible image data compression using the S-transform and Lemple-Ziv coding", Proc. SPIE, vol. 1091, pp. 2-18, 1989

[Bruekers1] F. Bruekers, A. Enden, "New networks for perfect inversion and perfect reconstruction", IEEE J. Sel. Areas Commun., vol.10, no.1. pp. 130-137, Jan. 1992.

[Calderbank1] A. R. Calderbank, I. Daubechies, W. Sweldens, B-L. Yeo, "Wavelet transforms that map integers to integers", Applied and Computational Harmonic Analysis 5, pp. 332-369 (1998) article No HA979238.

[Claypoole1] R. L. Claypoole Jr., R. G. Baraniuk, R. D. Nowak "Adaptive wavelet transforms via lifting," Proc. ICASSP'98, pp. 1513-1513, May 1998.

[Cornog1] K. Cornog, "Factors in preserving video quality in post-production when cascading compressed video system" 137'th SMPTE Technical Conference in New Orleans, September 7, 1995, paper no. 137-26.

[Daubechies1] I. Daubechies, W. Sweldens, "Factoring wavelet transforms into lifting steps," J. Fourier Anal. Appl., 4 (3), pp. 247-269, 1998.

[Dewitte1] S. Dewitte, J. Cornelis "Lossless integer wavelet transform", IEEE Signal Processing Letters, vol. 4, no. 6, pp. 158-160, June 1997.

[Gouze1] A. Gouze, M. Antonini, M. Barlaud "Quincunx lifting scheme for lossy image compression," Proc. ICIP'00, pp. 665-668, September 2000.

[Hao] P. Hao, Q. Shi, "Matrix factorizations for reversible integer mapping," IEEE Transactions on Signal Processing, vol. 49, pp. 2314-2324, October 2001

[Heer1] V. K. Heer, H-E Reinfelder, “A comparison of reversible methods for data compression”, Proc. SPIE, vol. 1233, pp. 354-365, 1990.

[Horne1] C. Horne, T. Naveen, A. Tabatabai, R. O. Eifrig, A. Luthra, “Study of the characteristics of the MPEG2 4:2:2 profile – application of MPEG2 in studio environment”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 6, no. 3, June 1996.

[Komatsu1] K. Komatsu, K. Sezaki, "Reversible discrete cosine transform," Proc. ICASSP'98, pp. 1769 - 1772, May 1998.

[Komatsu2] K. Komatsu, K. Sezaki, "Design of lossless LOT and its performance evaluation," Proc. ICASSP'00, pp. 2119 - 2122, June 2000.

[Komatsu3] K. Komatsu and K. Sezaki, "2D lossless discrete cosine transform," Proc. ICIP'01, pp. 466 - 469, October 2001.

[Komatsu4] K. Komatsu, K. Sezaki, "Lossless 2D discrete Walsh-Hadamard transform," Proc. ICASSP'01, pp. 1917 - 1920, May 2001.

[McDarby1] G. McDarby, P. Curran, C. Heneghan, and B. Celler, "Necessary conditions on the lifting scheme for existence of wavelets in $L_2(R)$," Proc. ICASSP'00, pp. 524 - 527, June 2000.

[Said1] A. Said, W. Pearlman, “An image multiresolution representation for lossless and lossy compression”, IEEE Transactions on Image Processing, vol. 5, no. 9, pp. 1303-1310, September 1996.

[Said2] A. Said, W. Pearlman, “Reversible image compression via multiresolution representation and predictive coding,”, Proc. SPIE, vol. 2094, pp. 664-74, 1993

[Sweldens1] W. Sweldens “The lifting scheme: a custom-design construction of biorthogonal wavelets” Applied and Computational Harmonic Analysis 3, 186-200 (1996) article no. 0015.

[Tabatabai] D. Le Gall and A. Tabatabai, "Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques," Proc. ICASSP'88, pp. 761 - 764, April 1988.

[Taubman1] D. Taubman, "High performance scalable image compression with EBCOT", Proc. ICIP 99, pp. 344-348, October 1999.

[Taubman2] D. Taubman, "High performance scalable image compression with EBCOT," IEEE Trans. Image Processing, vol. 9, pp. 1158 - 1170, July 2000.

[Wu1] X. Wu, M. Memnon, "Context-based, adaptive, lossless image coding," IEEE Trans. Commun. 45, pp. 437-444, 1997.

[Wu2] X. Wu, M. Memnon, "Lossless compression of continuous-tone images via context selection, quantization, and modeling," IEEE Trans. Image Processing, vol. 6, pp. 656-664, 1997.

[Zandi1] Zandi, M. Boliek, E. L. Schwartz, A. Keith. "Compression with reversible embeded wavelets with an enhanced binary mode," Proc. ICASSP'96, pp. 1963-1966, May 1996

[Zandi2] Zandi, J. Allen, E. Schwartz, M. Boliek, "CREW: compression with reversible embedded wavelets," in Proc. IEEE Data Compression Conf., J. A. Storer and M. Cohn, Eds., Snowbird, UT, pp. 212-221, March 1995

[Zeng] Y. Zeng, G. Bi, Z. Lin "Integer sinusoidal transforms based on lifting factorization," Proc. ICASSP'01, pp. 1181-1184, May 2001.

Author's contributions

[Domański1] M. Domański, K. Rakowski, "Color transformations for lossless image compression", Proceedings of X European Signal Processing Conference, EUSIPCO 2000, Tampere, Vol. III, pp. 1361-1364.

[Domański2] M. Domański, K. Rakowski, "A simple technique for near-lossless coding of color images" Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS 2000, Geneva, vol. III, pp. 299 – 302

[Domański3] M. Domański, K. Rakowski, "Near-lossless color image compression with no error accumulation in multiple coding cycles", Proceedings of 9th International Conference on Computer Analysis of Images and Patterns, CAIP 2001, Warszawa, Springer Verlag Lecture Notes in Computer Science Volume 2124, Issue, pp. 85-91.

[Domański4] M. Domański, K. Rakowski, "Lossless and near-lossless image compression with color transformations" Proceedings of IEEE International Conference of Image Processing, ICIP 2001, Thessaloniki, vol. III, pp. 454-457.

[Domański5] M. Domański, K. Rakowski, "Near-lossless color image compression for multimedia applications", Proceedings of EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services, ECMCS 2001, Budapest, pp. 181-184

[Rakowski] K. Rakowski, M. Domański, "Experimental analysis in multiple image coding-decoding cycles with color transformations," Proceedings of 10th International Workshop on Systems Signals and Image Processing, IWSSIP'2003, Prague, pp. 69-72.

Standards

[H261] ITU-T Recommendation H.261 – p×64 kbits/sec video coding standard for audiovisual services . 1993.

[H263] ITU-T, "Video Coding for Narrow Telecommunication Channels at < 64kbit/s", Recommendation H.263, 1996.

[JPEG_1] ISO/IEC IS 10918-1 / ITU-T Rec. T.81, "Information technology – digital compression and coding of continuous-tone still images: requirements and guideline",

[JPEG_2] ISO/IEC IS 10918-2 / ITU-T Rec. T.83, "Information technology – digital compression and coding of continuous-tone still images: compliance testing",

[JPEG_3] ISO/IEC IS 10918-3 / ITU-T Rec. T.84, "Information technology – digital compression and coding of continuous-tone still images: extensions",

[JPEG_LS] ISO/IEC JTC 1/SC 29/WG 1, "JPEG LS image coding system," ISO Working Document ISO/IEC JTC1/SC29/WG1 N399-WD14495, July, 1996.

[JPEG2000] ISO/IEC JTC 1/SC 29/WG 1, ISO/IEC FCD 15444-1: Information technology – JPEG 2000 image coding system: Core coding system, WG 1 N 1646, March 2000.

[MPEG1] ISO/IEC 11172-2, "Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbits/sec - Part 2: Video", 1993.

[MPEG2] ISO/IEC IS 13818-2 / ITU-T Rec. H.262, "Generic coding of moving pictures and associated audio, part 2: video", November 1994.

Books

[Bovik] A. Bovik "Image and video processing", San Diego, Academic Press, 2000.

[Gray] A. Gersho, R. M. Gray, "Vector quantization and signal compression," Norwell, Kluver Academic Press 1991.

[Jajant] N. Jajant, P. Noll, "Digital coiding of waveforms," Engelwood Cliffs NJ, Prentice Hall 1984.

[Marcelin] D. S. Taubman, M. W. Marcellin, "JPEG2000 image compression fundamentals, standards and practice" Kluwer Academic Publishers 2002.

[Ohm] J. R. Ohm, "Multimedia communication technology," Springer 2004.

[Pennebaker] W. B. Pennebaker, J. L. Mitchell, "JPEG still image compression standard," New York, Van Nostrand Reinhold 1993.

[Graham] R. L. Graham, D. E. Knuth, O. Patashnik, "Concrete mathematics. A foundation for computer science," Addison-Wesley Publishing Company, 1994.