VĚDECKÉ SPISY VYSOKÉHO UČENÍ TECHNICKÉHO V BRNĚ

Edice PhD Thesis, sv. 331

ISSN 1213-4198

Ing. Jindřich Nový

Digital Filtering and Compression in Image Processing and Volume Rendering

Brno University of Technology
Faculty of Mechanical Engineering
Institute of Mathematics

Ing. Jindřich Nový

DIGITAL FILTERING AND COMPRESSION IN IMAGE PROCESSING AND VOLUME RENDERING

DIGITÁLNÍ FILTROVÁNÍ A KOMPRESE V ANALÝZE OBRAZU A OBJEMOVÉM RENDERINGU

Short version of Ph.D. Thesis

Study Field: Mathematics
Supervisor: Prof. RNDr. Miloslav Druckmüller, CSc.
Opponents: Doc. PaedDr. Dalibor Martišek, Ph.D.
Doc. RNDr. Bedřich Půža, CSc.
Presentation Date: 15.6.2005

Keywords: volume rendering, raycasting, convolution, FFT, orthogonal wavelet, biorthogonal wavelet, DCT, BWT, RLE, Huffman coding, Shannon-Fano coding, arithmetic coding, KLT, RGB, CMY, YUV, HSV, black body, Visible Human Project

Klíčová slova: objemový rendering, raycasting, konvoluce, FFT, ortogonální wavelety, biortogonální wavelety, DCT, BWT, RLE, Huffmanovo kódování, Shannon-Fanovo kódování, aritmetické kódování, KLT, RGB, CMY, YUV, HSV, absolutně černé těleso, Visible Human Project

The original of the dissertation is available in the department of Studies of the Faculty of Mechanical Engineering, Brno University of Technology, Technická 2, Brno, Czech Republic.

ISBN 80-214-2983-6
ISSN 1213-4198

Contents

Introduction

The purpose of the dissertation is to introduce a new approach of general volume data analysis. Many recent measurements in engineering practise lead to a three dimensional volume representation of results which are quite hard to be analyzed in an image representation since its true nature is three dimensional. As a particular example can be noted data acquired by a confocal microscope or nuclear magnetic resonance scanners which are of great importance in medicine and biology. Because of the fact the volume data are stored in a general three dimensional matrix it can be used for analysis of any volume representation of a physical phenomenon such as results of numerical solution of differential equations etc. Therefore it is not limited only to measured datasets.

The visualization method is designed especially to simulate a human perception of observing a volume object, particularly linear perspective projection, visibility based on raycasting and color combining techniques are used. Thus one can deduce features of a volume object naturally without a proper knowledge of these methods. An important property of this visualization method is that a volume object can be observed interactively so a factual emphasis in the dissertation is aimed to render a visualization in the fastest possible time, but without a hardware acceleration, because recent accelerators don't provide sufficient support for high quality accelerated volume rendering.

The next part of the dissertation is dedicated to principles and application of digital filters to image and volumetric data, where also methods of adaptive filtering such as adaptive kernel convolution are presented with particular examples of their

use in practise. Volume analysis unfortunately brings lots of problems to solve. The most apparent problem is a vast amount of storage space required to contain all the information contained in a volume object in contrast with a less expensive 2D image representation. It is essential to overcome this problem since rather small volume data can reach beyond a capacity of todays computers. To reduce a total amount of information there are proposed two types of compression algorithms. The first, lossless algorithm is based on Burrows-Wheeler block sorting and adaptive arithmetic encoding with finite context prediction. The second is aimed to lossy compression, which is based on progressive DCT encoding.

Chapter 1

Digital Color Representation

1.1 Introduction

The most popular part of image processing is processing of color images. There exist various ways to store and interpret color information present in such images. A color is often represented by three coordinates which can be imagined as a spatial vector. Thus a color can be represented by a vector in so called **color space**. Further will be discussed properties of the most frequent color spaces in order to learn how a color information can be stored and which color space is suitable for a patricular processing. More in-depth information about color spaces can be found in [7].

1.2 CIE Chromacity Diagram

The name CIE comes from french **Comission Internationale de l'Éclairage** and means International Commission on Illumination. It is the international authority on light, illumination, color and color spaces. The standard reference for color definition is CIE chromacity diagram. This diagram was developed in 1931 and in full plot it is three dimensional with **tristimulus** X, Y, Z coordinates. The coordinates are linear measures of light power. The Y coordinate is called "luminance" which can

be regarded as a measure of percepted brightness. For better operating it was transformed to 2D x, y diagram where y axis shows color intensity and x axis chrominance of CIE-1931 standard observer as illustrated by the transformation equations:

$$x = \frac{X}{X+Y+Z}, \qquad y = \frac{Y}{X+Y+Z}. \tag{1.1}$$

The x, y coordinates are known as **chromacity** coordinates. The CIE diagram is based on visual response of this observer and is determined by physiological measurements of human color vision. Upper rounded boundary of the diagram is composed of pure spectral colors and their purety (saturation) decreases down to a white point in the center. Lower straight purple part is not a spectral color and is used as arbitrarily selected wavelength cut-off. Most frequently the color properties are described by red, green and blue x, y coordinates, which are called **primary illuminants** and so called **white point**. The recommended [2] CIE D_{65} 'daylight' whitepoint is:

$$x_w = 0.3127, \qquad y_w = 0.3290, \tag{1.2}$$

and the respective primary illuminants are:

$$\begin{pmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \end{pmatrix} = \begin{pmatrix} 0.64 & 0.30 & 0.15 \\ 0.33 & 0.60 & 0.06 \end{pmatrix}. \tag{1.3}$$

The D_{65} primary illuminants are intended to represent 'daylight' temperatures at various correlated color temperatures, which are colors that in a well defined sence corresponds to black body color at certain temperature. The correlated color temperature for D_{65} corresponds to roughly $6500K$[1].

1.3 Black Body and Sun Color

The effective sun temperature is about $5780K$. It's hard to determine its color since the sun is not a perfect black body and the final percepted color is also affected by

[1] About 6504K since the later more proper evaluation of hc/k fundamental constats ratio in the Planck formula.

atmospheric effects. However not a big mistake will be made by an assumption that irradiation of the sun is a perfect black body. We can calculate a color of it rather easily then using the **Planck irradiation law**:

$$P(\lambda, T) = \frac{4\pi^2 \hbar c^2}{\left(e^{\frac{hc}{k\lambda T}} - 1\right)\lambda^5}, \tag{1.4}$$

what tells what's the probability of emittance of photons at a particular wavelength λ for a black body with known temperature T. As a color percepted by human for a particular wavelength of light is known from the CIE diagram, it's possible to calculate black body color for a particular temperature as:

$$\vec{c}(T) = \frac{1}{P_T} \int_{\lambda_1}^{\lambda_2} P(\lambda, T)\vec{s}(\lambda) d\lambda, \tag{1.5}$$

where $\vec{s}(\lambda)$ is R,G,B color vector and $\vec{c}(T)$ is the final color of black body at temperature T,

$$P_T = \int_{\lambda_1}^{\lambda_2} P(\lambda, T) d\lambda \tag{1.6}$$

and the range of visible color wavelengths is $\Delta\lambda = \langle \lambda_1, \lambda_2 \rangle = \langle 450, 800 \rangle$ nm. The result for this calculation can be seen at fig.1.1, where the color assigned to the sun effective temperature is roughly white and turns to red for lower and to cyan for higher temperatures.

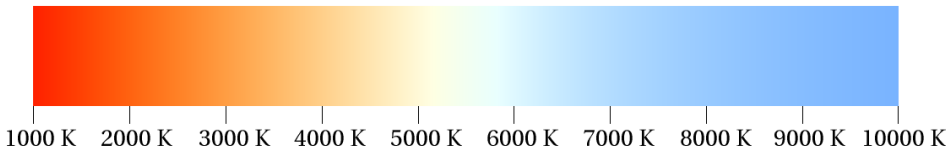


Figure 1.1: Approximate black body color at various temperatures.

Chapter 2

Digital filters in image processing

2.1 What a digital filter is?

First it's required to properly define a concept of a filter which is essential to understand methods described further. It is important to point out at the very beginning of this chapter that we are going to be involved exclusively in analysis of **digital signals**. Thus it's assumed that an input signal to be processed is sampled at a given sampling rate and quantized in a way that every sample has a finite bit depth which is the same for all the samples within the signal. Later, if we talk about a signal we mean a digital signal when not otherwise stated. By **digital filtering**[1] is meant processing samples of an input signal in the way that some arithmetics is performed to its samples to yield an output signal. Filtering methods are described in the following chapters.

2.2 Classification of Filters

Digital filters are often [8] divided in two cathegories dependent on the fact whether the filter can or cannot be considered as a linear system. A system is called linear if it has properties of **homogenity** and **additivity**. If one of these assumptions are

[1] Further abbreviated only to "filtering".

not true, the system is nonlinear. To demonstrate these and other filter properties let's have a filter $\mathcal{F}$ to process an input signal $a(t)$ to output signal $b(t)$, then:

- Filter $\mathcal{F}$ is called **homogenous** if amplitude change in a causes the same amplitude change in b:

$$\mathcal{F}\big[a(t)\big] = b(t) \implies \mathcal{F}\big[ka(t)\big] = kb(t) \qquad \forall t, \tag{2.1}$$

  where k is a constant.

- Filter $\mathcal{F}$ is called **additive** if processing of addition of two input signals a_1 and a_2 result in addition of two output signals b_1 and b_2:

$$\mathcal{F}\big[a_1(t)\big] = b_1(t),\ \mathcal{F}\big[a_2(t)\big] = b_2(t) \implies \mathcal{F}\big[a_1(t) + a_2(t)\big] = b_1(t) + b_2(t) \quad \forall t. \tag{2.2}$$

- Filter $\mathcal{F}$ is called **shift invariant** if shift in a causes identical shift in b:

$$\mathcal{F}\big[a(t)\big] = b(t) \implies \mathcal{F}\big[a(t+dt)\big] = b(t+dt) \qquad \forall t. \tag{2.3}$$

The last property of shift invariance is not required to consider a system linear, but most of linear and even nonlinear filters have this property.

2.3 Discrete Convolution Filters

A nice example of linear filter is a convolution filter. A discrete convolution filter can be defined as:

$$b(t) = \mathcal{F}_{h(k)}\big[a(t)\big] = a(t) * h(k) = \sum_{k=-K/2}^{K/2-1} a(t-k)h(k) \quad \forall t \in \{0, 1, ..., N-1\}, \tag{2.4}$$

where $h(k)$ is **convolution kernel** of K samples and $a(t)$ is a signal to be convolved. The $h(k)$ is also called **impulse response function** or **point spread function** in image processing. In this text is considered only the **finite impulse response**

(FIR) or **truncated infinite impulse response (TIIR)** function $h(k)$ exclusively so that K is set to some sufficiently large finite value. The name of $h(k)$ is a finite impulse response because if an impulse is passed, i.e. $a(t = t_0) = 1$, zero otherwise, through the convolution filter, function $h(k)$ shifted of t_0 is obtained as the output. The sum of all samples within the kernel $h(k)$ must be a non-zero constant in case of low-pass filter what is fulfilled if it's normalized, so the condition 2.5 must hold.

$$\sum_{k=0}^{K-1} h(k) = 1. \tag{2.5}$$

An analogous condition 2.6 must hold for a high-pass filter:

$$\sum_{k=0}^{K-1} g(k) = 0. \tag{2.6}$$

If we look at (2.4) the argument of a can reach below zero and above the total samples N in the signal. We can't neglect this boundary problem because signal of a finite lenght is processed. By a style how the boundary problem is solved can be distinguished various types of discrete convolution.

2.4 Adaptive Kernel Convolution

It is appropriate for some signals to change shape of kernel during convolution. If the idea is improved a little bit, we can say that we needn't only change $h(k)$ at the end and beginning of $a(t)$ but also let $h(k)$ be somehow dependent on contents of $a(t)$ itself. We call it **adaptive kernel convolution** when this approach is used.

One application of this type of convolution is demonstrated here which is very suitable for error elimination from images. The divide and conquer strategy is followed here to fulfill this task, so the method itself consists of not one but a few simple steps. The calculation consists of three steps, where adaptive kernel convolution is used in the end:

1. Low-pass filtering of the original image $I_1(x, y)$ to obtain processed image $I_2(x, y)$.

2. Detection of defect pixels based on difference checking between I_1 and I_2.

3. Replacing of defect pixels and their interpolation using acyclic adaptive kernel convolution.

For a given image $I_1(x, y)$ 2D discrete acyclic convolution with symmetric TIIR gaussian kernel of K samples is calculated in each dimension to obtain I_2:

$$I_2(x,y) = \mathcal{F}_{h(k_x,k_y)}\big[I_1(x,y)\big] = I_1(x,y) * h_1(k_x,k_y) = I_1(x,y) * \Big[e^{-\frac{k_x^2+k_y^2}{\sigma_1^2}}\Big]_K \tag{2.7}$$

where K is large enough to represent the point spread function $h_1(k_x, k_y)$. The next step is to mark defective pixels. An information about defective pixels is held in error matrix $E(x, y)$, which is calculated by using the following bad pixel criterion:

$$E(x,y) = \begin{cases} 0 & \text{if } |I_1(x,y) - I_2(x,y)| \leq I_t; \\ 1 & \text{otherwise.} \end{cases} \tag{2.8}$$

where I_t is appropriatelly selected threshold. When $E(x, y)$ is calculated, we perform adaptive kernel convolution (AKC) of $I_1(x, y)$ for every $E(x, y) = 1$ in order to interpolate bad pixel in $I_1(x, y)$. Even if it's a 2D problem (and might be 3D problem for volume data) it can be reduced to one dimensional AKC, what can be done by creating a vector function $\vec{p}(n)$ which contains coordinates of neighbouring pixels in not decreasing distance order with increasing n. It is shown in the fig.2.1. The filtered image is then calculated by AKC as:

$$I_1(x,y) = \frac{1}{||h_2||} \sum_{n=1}^{N-1} I_1\big[(x,y) - \vec{p}(n)\big] h_2(n,x,y) \qquad \forall E(x,y) = 1, \tag{2.9}$$

where

$$h_2(n,x,y) = \begin{cases} e^{-\frac{|\vec{p}(n)|^2}{\sigma_2^2}} & \text{if } E\big[(x,y) + \vec{p}(n)\big] = 0; \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

is adaptive kernel constructed from TIIR gaussian kernel which expresses decreasing weight of convolved pixel with increasing distance $|\vec{p}(n)|$. The norm $||h_2||$ is calculated during AKC as:

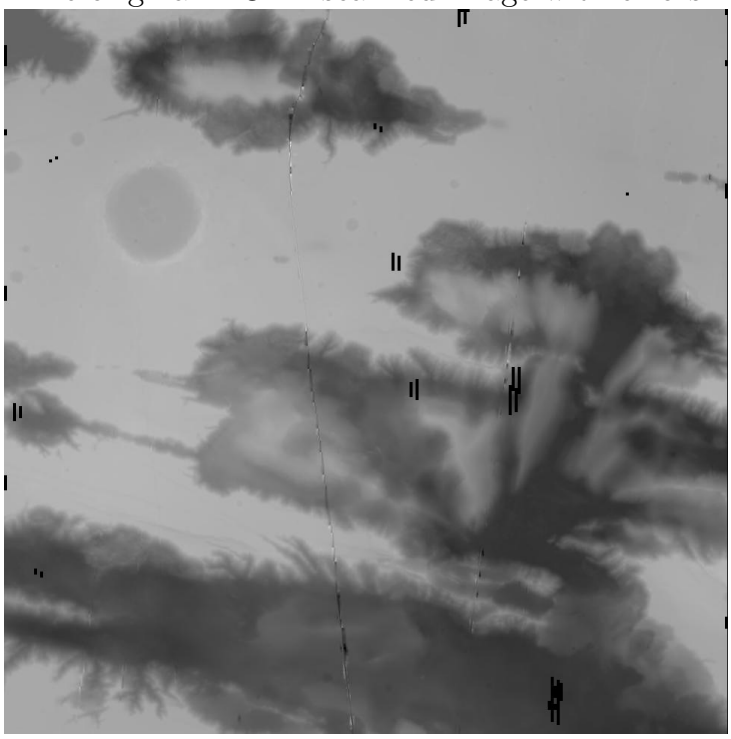
$$||h_2|| = \sum_{n=1}^{N-1} h_2(n,x,y). \tag{2.11}$$

151	129	101	89	81	69	82	90	102	130	152
131	97	70	61	49	45	50	62	71	98	132
103	72	57	37	29	25	30	38	58	73	104
91	63	39	21	13	9	14	22	40	64	92
83	51	31	15	5	1	6	16	32	52	84
74	46	26	10	2	0	3	11	27	47	75
85	53	33	17	7	4	8	18	34	54	86
93	65	41	23	19	12	20	24	42	66	94
105	76	59	43	35	28	36	44	60	77	106
133	99	78	67	55	48	56	68	79	100	134
157	135	107	95	87	80	88	96	108	136	158

Table 2.1: Pixel ordering n in non-decreasing distance from the center (Euclid metric).

Note that the norm is also dependent on position x, y by equation 2.10.

The original MOLA scanned image with errors.



Filtered by adaptive kernel convolution.

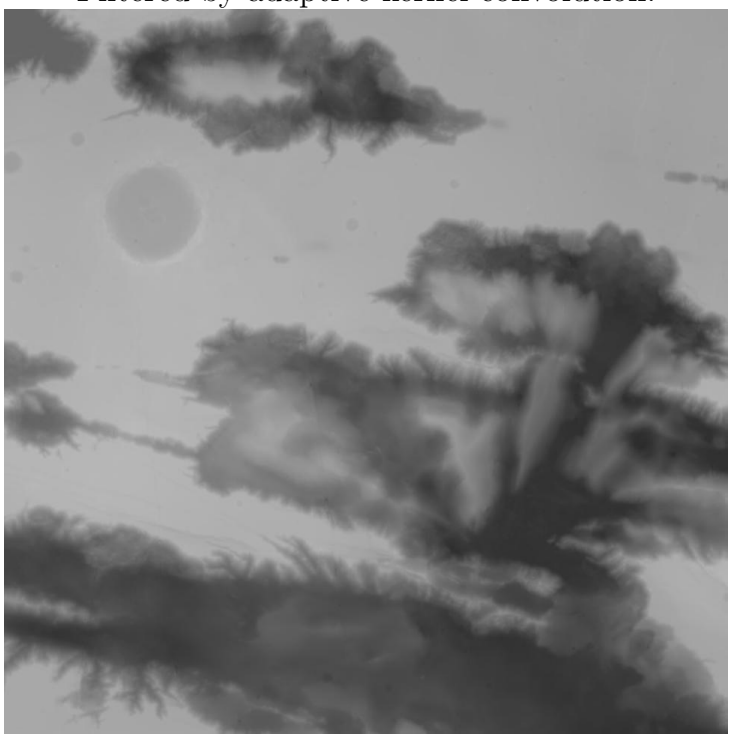


Figure 2.1: Usage of the adaptive kernel convolution.

Chapter 3

Volume rendering

Analysis of volume data is quite complicated task to solve. There exist several approaches to do such analysis. The basic one is to decompose a 3D object to slices and to analyze them as usual images. This method allows us to look inside the volume of 3D object but seems to be unusable when our demand is to observe the object in a different direction. Then a 3D visualization technique is likely to be used. The basic methods of such 3D visualization are based on extraction of polygonal meshes describing isosurfaces[1]. One can enjoy a 3D impression when observing such isosurface but a disadvantage of it is that the isosurface is in fact two dimensional so one has to extract more such isosurfaces to predict how a property of 3D object changes in volume. Furthermore, algorithms of isosurface extraction are rather complicated and often erroneous for a complex 3D objects. Our approach is to employ a volume rendering technique, which is intended to consider volume information as native 3D. Unfortunately this technique is very computationaly expensive so optimizations and speedup is of our particular importance. The purpose of this chapter is to develop a volume rendering method, apply it to large scale VHP volume dataset and conclude with discussion.

[1] A surface on which a feature of the 3D object is (almost) constant.

3.1 Principles of Volumetric Raycasting

In order to develop a suitable technique of volume rendering we must define how the 3D object will actualy be analyzed. First of all we want to analyze a 3D object from various points of view and directions. Therefore a position of an observer and his viewing orintation has to be specified. Then, because of the fact we'll display an image of the 3D object on a 2D screen, we have to use some method of projection. Since a human perception of observing a 3D scene is similar to linear perspective projection we'll use such projection to let the result look natural to a human observer. Finaly, because we have a volumetric object, it is good idea to have a possibility to "see in depth", i.e. to model translucency of the data.

3.1.1 What Raycasting is

The basic idea of the raycasting is to cast a ray through a volume object for every single pixel on the screen and trace properties of the volume object along the ray in order to return a single pixel value per ray to describe volume properties of the object. The situation is shown on fig.3.1, where position and orientation of the

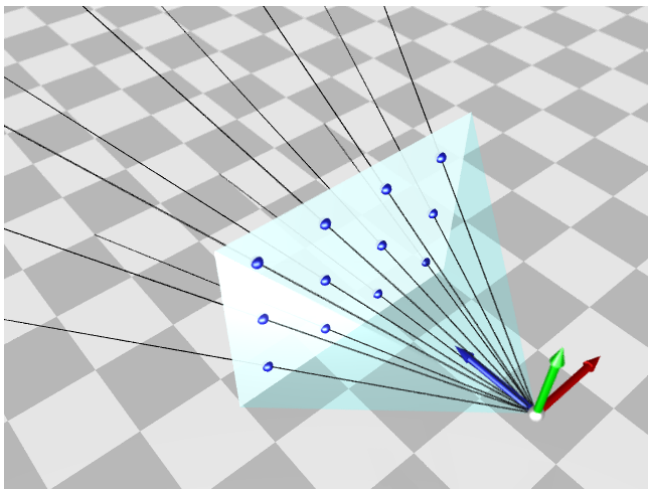


Figure 3.1: Illustration of rays cast from the position of observer.

observer is defined by the perspective pyramid and rays are colored black with the

tracing direction away from the observer. Note that in the fig.3.1 is demonstrated creating of raycasted projection to a screen of 4×3 pixel resolution which is mapped to the perspective plane and rays passes through the center of their corresponding pixels, what is denoted by small blue spheres in the image.

3.2 Volume Renderer Design

The developed volume rendering software saves all the volume data in compressed form in the way that the volume data is decomposed into cubes[2] so that only this part of volume is needed to be stored in memory when a ray passes through it. Frequently used cubes are stored in memory in order to prevent a need to decompress them when a ray penetrates to another cube that was traced recently. The software allows to use only a memory specified by an user. To keep only last recently used cubes the LRU [6] algorithm is used. The software uses progressive rendering in the way that at first only a fraction of total rays is cast into a 3D scenery. After calculation of resultant colors, these are interpolated for pixels for which no rays have been cast and shown on the screen. This allows an user to see a temporary result of rendering even if rendering goes on. The pixel values are calculated in a way that pixel intensities are gathered along a ray in a given[3] precision so that the result is a vector of intesities $\vec{V_c}$ of arbitrary size for each channel c in the volume dataset. The $\vec{V_c}$ can be processed in various ways to obtain a single pixel value to be displayed on the screen. At first a convolution with thin gaussian TIIR kernel is calculated to suppress sharp intensity jumps and similar artifacts caused by discontinuities between voxels:

$$\sum_{k=-K/2}^{K/2-1} \vec{V_c}(i-k)h(k), \qquad \forall i, c \tag{3.1}$$

then opacity tracing algorithm is used. Description of such techniques can be found in [5], [3]. The opacity tracing algorithm works in the way that a certain small amount

[2] of $8 \times 8 \times 8$ up to $32 \times 32 \times 32$ voxels each.

[3] mostly sub–voxel

of the first intensities along the ray are ignored in order to skip low–intensity layer around a volume object. Then the final pixel intensity of c–th channel of the volume data is calculated as

$$I_c = \frac{1}{N_2 - N_1} \sum_{i=N_1}^{N_2} T^{i-N_1} \vec{V}_c(i), \qquad \forall c \tag{3.2}$$

where T is translucency coefficient[4], N_1 is the index in $\vec{V}_c$ for the first not ignored intensity value, N_2 is an index in $\vec{V}_c$, where the contribution to the final pixel intesity is neglectable due to small value of T^{i-N_1} in 3.2. Then the final color $\vec{P}$ in RGB space of the pixel displayed on screen is calculated like

$$\vec{P} = \left\| \sum_{c=0}^{C} \vec{P}_c I_c \right\|, \tag{3.3}$$

where C is the cotal number of channels in volume data, P_c is normalized base color of the channel in RGB color space. Normalization in 3.3 denotes normalization and/or saturation of the RGB components in $\vec{P}$.

[4] $T \in \langle 0, 1 \rangle$, where for $T = 1$ is the object completely translucent

Chapter 4

Lossy Image Compression

4.1 Discrete Cosine Transform

Another method of image decorrelation is the Discrete Cosine Transform (DCT). In numerous cases DCT is used frequently in practice since there exist lots of fast algorithms which calculates DCT decorrelation much faster with results close to the KLT. Many image compression algorithms are based on DCT, for instance JPEG [9]. There exist various types of DCT which are designed to fast image decorrelation in separate image fragments or by partially overlapped fragments[1]. The most frequent DCT type is DCT-II, which is in 1D defined as

$$F(k) = C(k) \sum_{x=0}^{X-1} f(x) \cos \frac{k(2x+1)\pi}{2X}, \tag{4.1}$$

where X is total number of samples in $f(x)$, and

$$C(k) = \begin{cases} \sqrt{\frac{2}{N}} & \text{if } k \neq 0; \\ \sqrt{\frac{1}{N}} & \text{otherwise.} \end{cases} \tag{4.2}$$

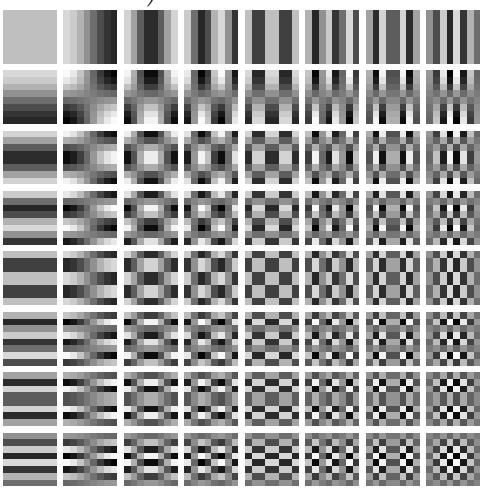
[1] So called Lapped Orthogonal Transform (LOT).

In our case we'll use 2D DCT-II, where we can also compose it from row and column 1D transforms. Then we can write the 2D DCT-II as

$$F(k_x, k_y) = C(k_x)C(k_y) \sum_{y=0}^{Y-1} \left[\sum_{x=0}^{X-1} f(x, y) \cos \frac{k_x(2x+1)\pi}{2X} \right] \cos \frac{k_y(2y+1)\pi}{2Y}. \quad (4.3)$$

As we have mentioned, the DCT transform is used because its calculation speed. It is so because close similarity of DCT basis vectors with their optimal ones for a natural picture. These discrete bases are calculated as a spatial response to impulse in KLT or DCT[2] domain. We can interpret the shape of basis vectors shown at fig.4.1a as an information of what type of detail from an input image contains a single DCT-II basis coefficient. In particular, we can say that the most of horizontal information contain the leftmost coefficients, the highest details the coefficient in bottom right, etc.

a) natural order



b) variance descending (zig-zag) order

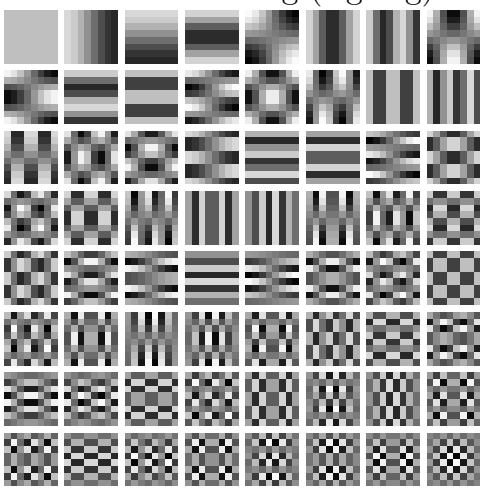


Figure 4.1: DCT basis vectors for 8×8 transform.

[2]Fig.4.1.

4.2 DCT–based Lossy Compression Algorithm Proposal

The purpose of all compression algorithms in image compression is to reduce the information capacity up to its minimum in the fastest possible time at minimal memory requirements. Furthermore, the quality loss of decompressed image should be minimal. Unfortunately, compromises have to be made because these optimum criteria are dependent on each other. In particular, the most of recent lossy image compression algorithms are based on the DCT (section 4.1) even if it's not optimal, because the calculation time is much shorter in comparison with KLT.

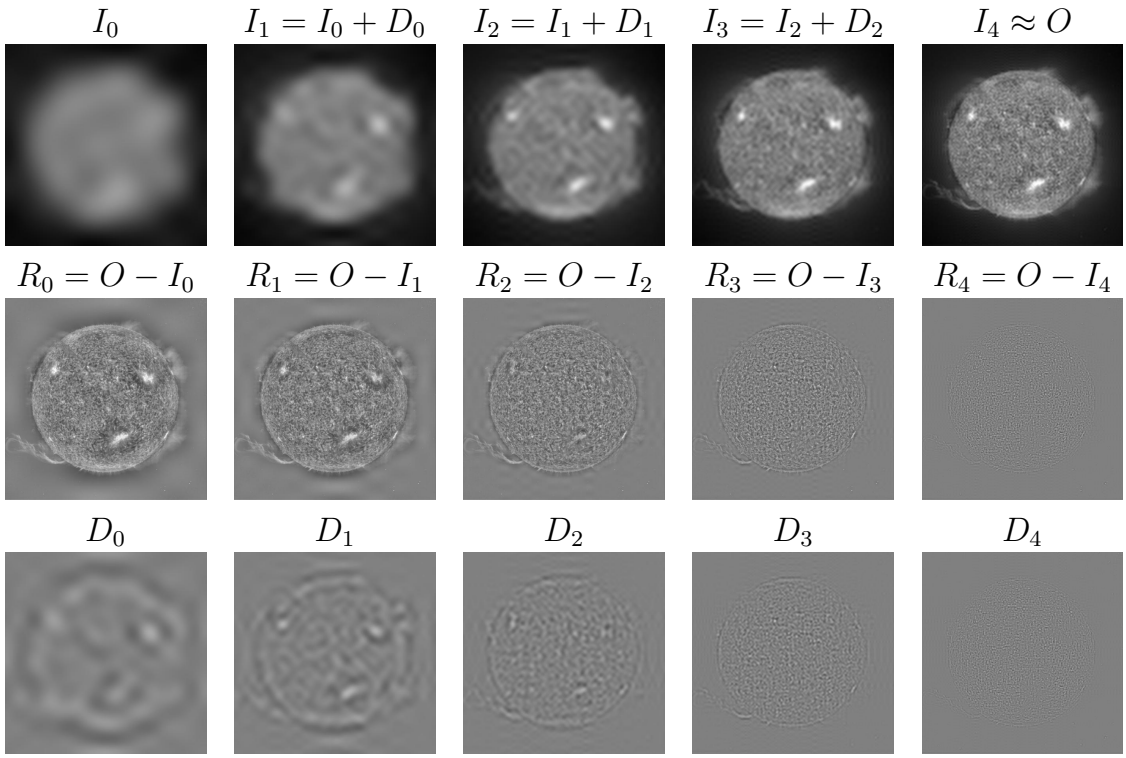


Figure 4.2: Progressive 8×8 encoding with zero-padded 2D IDCT prediction.

The preferred scheme for the compression algorithm is progressive. It means that an original image O to be compressed is stored at various levels of detail and only

differences are encoded. The progressive scheme was chosen because of error resistence and possibility of displaying preview at various scales of image even from its small loaded context. The encoding itself is based on 2D DCT but is designed to be easily modified to KLT. The original image O is decimated to a size to fit one single 2D DCT fragment and saved to output file[3]. Then the fragment is upsampled to the original size of O with suitable technique of interpolation. After it the upsampled image I_0 (see fig.4.2 where the fragment size is 8×8 pixels) is subtracted from O which yields R_0. That was the first step and the following steps are calculated iteratively in the way that the number of DCT fragments is increased by factor of 4^4 per iteration what ensures convergence to the original image:

$$\begin{aligned} D_i &= \uparrow \mathrm{DCT}\Big(\downarrow R_i\Big), \\ I_{i+1} &= I_i + D_i, \\ R_{i+1} &= O - I_{i+1}, \\ i &= i + 1, \end{aligned} \tag{4.4}$$

where $\downarrow$ denotes decimation and $\uparrow$ denotes interpolation. After each iteration D_i is encoded and saved to output file. The total number of iterations is equal to $b = \log_2 S$, where S is the largest dimension of image, where $S \leq 2^b$. This method gives better or comparable compression in comparison with recent lossy image compression formats with comparable signal-to-noise ratio.

[3]It means quantized, zig-zag ordered, RLE and entropy encoded.
[4]2 per dimension

Chapter 5

Lossless Image Compression

A different treatment to image compression has to be made in order to remove redundant information from input data with a request of lossless reconstruction of the original image. There exist a few approaches to lossless compression algorithms. The simplest approaches are RLE algorithms that replaces large sequences of identical characters with length-code pairs. The other classical methods used frequently are alphabet substitution approaches such as Shannon-Fano coding or Huffman [4] coding. These methods construct binary trees in order to assign the smallest bit sized code to the most probably used characters. To the other group belong dictionary or sequentional based methods from a wide Lempel-Ziv family such as LZ77 [10] or LZ78 [11]. These methods uses dynamically updated databases of used sequences, where a compression is achieved because a single code is sent out for every frequently used sequence. In our approach we use multiple level compression, where arithmetic coding scheme is used at the end, which achieves the best compression in general from all the mentioned methods.

The arithmetic coding is a bit similar to Huffman coding in the way that it assigns less bit-sized codes to the most probable characters. The principal difference is that in case of arithmetic coding the codes needn't be of integer bit length. In fact the output of arithmetic coder is one huge binary number which represents all characters present in message of any finite alphabet.

5.1 Lossless Compression Algorithm Proposal

The proposed algorithm for lossless image compression is based on Burrows-Wheeler block-sorting algorithm [1], [6], hierarchycal RLE coding and adaptive arithmetic coding with finite context prediction.

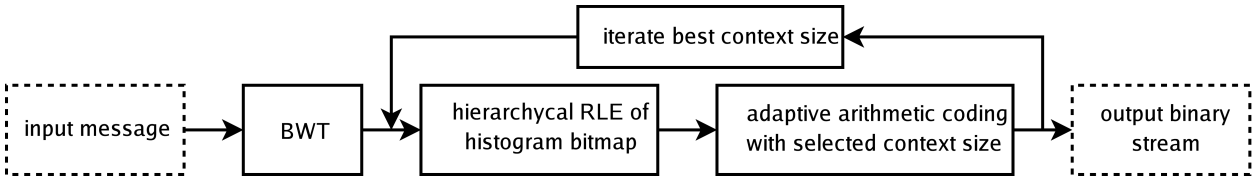


Figure 5.1: Scheme of the proposed compression algorithm.

The characters within a message are first reordered using the BWT to gather characters of similar contexts close together. Then the message is separated to blocks of specified size. Then for each block a histogram bitmap is calculated to reduce message alphabet states in the block passed to the entropy coder. The histogram bitmaps are encoded by the hierarchycal RLE encoding. Multiple iteration passes are then performed in order to find an optimal size of context to compress the message to a minimal size. The context starts at size of 256 characters and is doubled in each iteration up to 16384 characters. Usual range for the optimal context size is 1024 up to 8192 characters. The reason for doubling the context size is time expense of the compression. It could be found in a precission up to one character. The compression software supports up to 16bit per letter characters in a message. The purpose of the hierarchycal RLE coding in the compression scheme in table 5.1 is that for 16-bit characters we have to store histogram bitmap for 65536 characters, what is the bitmap of size 8192 bytes what is frequently reduced below 1/100 of its original size. In case of 8-bit characters in message, the histogram size is mostly halved. A comparison of effectivity of the proposed algorithm and other commercial and non-commercial lossless compression formats can be seen from table 5.1.

Name	File size [B]	Entropy [b]	Algorithm
GIF	700 206	5.342	LZW
TIFF	621 772	4.743	LZ77
GNU zip v1.3.3	608 016	4.638	LZ77
UNIX compress v4.2.4	604 225	4.609	LZW
RAR v3.3 beta 1	566 518	4.322	unknown
PNG	542 779	4.141	LZ77
bzip2 v1.0.2	470 925	3.592	BWT, Huffman
proposed algorithm	427 934	3.265	BWT, Arit.

Table 5.1: Results and comparison of the proposed compression algorithm.

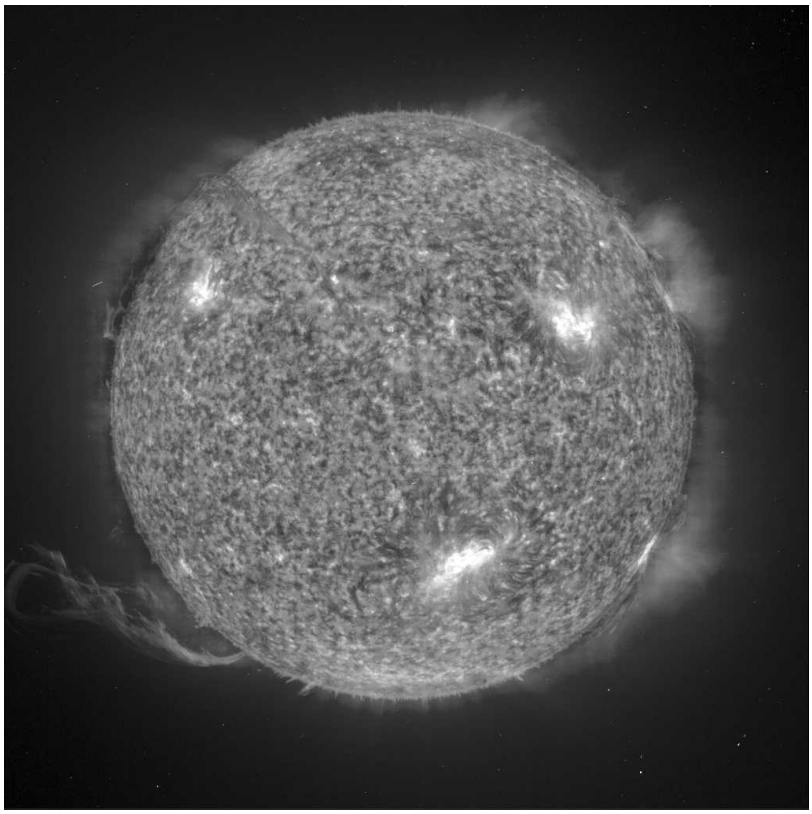
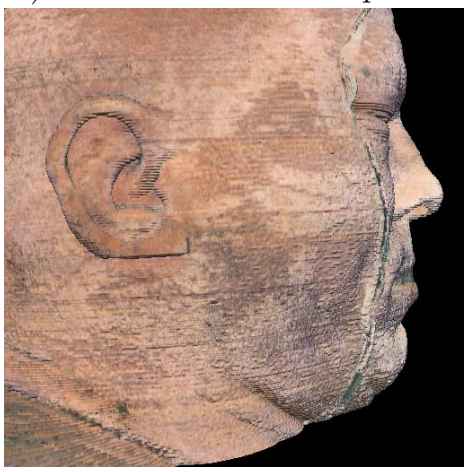


Figure 5.2: Sample 1024×1024 grayscale image used for compression.

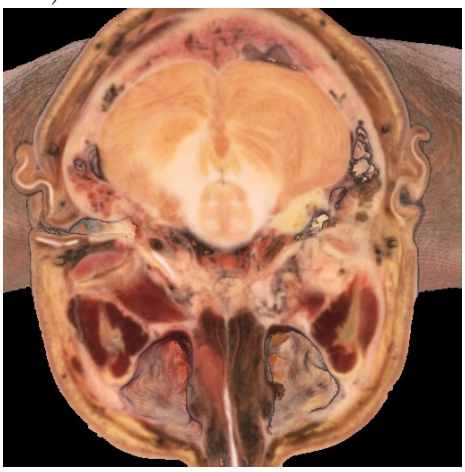
a) cross section in eyes area

b) face of the cadaver

c) cadaver viewed from profile



d) nose hollow and ear area



e) photography of Joseph Paul Jernigan

f) area with cervical, pith and vocal chord

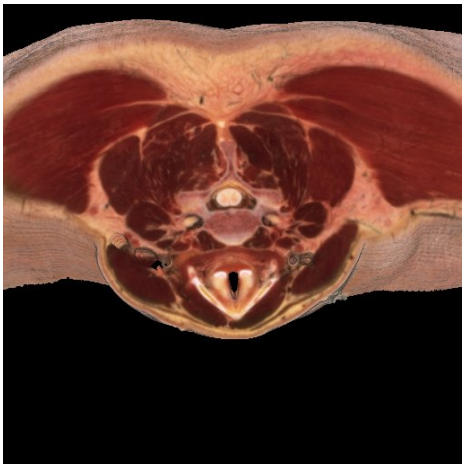


Figure 5.3: Application to the Visible Human Project.

Bibliography

[1] Burrows M., Wheeler D.J. *A Block-sorting Lossless Data Compression Algorithm*. Digital System Research Center, 1994.

[2] CCIR Rec.709. *Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange*. ITU-R Recommendation BT.709, 1990.

[3] Druckmüller M. *Opacity simulation technique for confocal microscope image processing*, vol. 1. Journal of Applied Biomedicine, 2003.

[4] Huffman D.A. *A method for the construction of minimum-redundancy codes*, vol. 40. Proceedings of the IRE, 1952.

[5] Martišek D. *The 2D and 3D Processing of Images Provided by Conventional Microscopes*, vol. 24. Scanning, 2002.

[6] Nový J. *Numerické metody rekonstrukce 3D modelu zemského povrchu*. ÚFI FSI VUT Brno - diplomová práce, 2002.

[7] Rogers D. F. *Procedural Elements for Computer Graphics*. McGraw-Hill Book Company, New York, 1985.

[8] Smith S. W. *The Sciencist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, P.O. Box 502407, San Diego, CA 92150-2407, 1999.

[9] Wallace G. K. *The JPEG still picture compression standard*. Communications of the ACM, 1991.

[10] Ziv, Lempel. *A universal algorithm for sequential data compression*, vol. 23. IEEE Transactions in Information Theory, 1977.

[11] Ziv, Lempel. *Compression of individual sequences via variable-rate coding*, vol. 24. IEEE Transactions in Information Theory, 1978.

Curriculum Vitae of Jindřich Nový

Education:

- 15.8.1979: born in Sokolov, Czech Republic.
- 1985-1993: basic school with extended language education.
- 1993-1997: secondary school – Private Bussiness Academy in Vyškov.
- 1997-2002: Brno University of Technology, Faculty of Mechanical Engineering, Department of Physical Engineering: MSc. in Physical Engineering in 2002.
- 2002-2005: Brno University of Technology, Faculty of Mechanical Engineering, Institute of Mathematics: Postgraduate doctoral studies.

Practise:

- 2001-2002: one semester study stay in Italy at University of Bari, Department of Plasma Chemistry – implementation of simulation software for investigation of ion-solid interactions, solid state phase changes and wave equations.
- 2003-2004: one semester study stay in Austria at TU Wien, Institute of Discrete Mathematics and Geometry – implementation of software for 3D visualization and filtration of noise from images acquired by satellite.
- 2004-2005: Hired by Red Hat Inc. as base OS programmer of Red Hat GNU/Linux operating systems. Since 2005 team leader of 12 base OS programmers.

Abstract

The purpose of the dissertation is to introduce a new approach of general volume data analysis and image processing of 2D images. The volume data visualization method is designed especially to simulate a human perception of observing a volume object, particularly linear perspective projection, visibility based on raycasting and color combining techniques are used. Thus one can deduce features of a volume object naturally without a proper knowledge of these methods. The next part of the dissertation is dedicated to principles and application of digital filters to image data where also methods of adaptive filtering such as adaptive kernel convolution are presented with particular example of its usage in practise. Then principles and filters using radix 2 FFT algorithm are presented. To reduce a total amount of information there are proposed two types of compression algorithms. The first, lossless algorithm is based on Burrows-Wheeler block sorting and adaptive arithmetic encoding with finite context prediction. The second is aimed to lossy compression, which is based on progressive DCT encoding. The principles and examples of orthogonal and biorthogonal wavelet transform application in image compression is also presented.

Keywords:
volume rendering, raycasting, convolution, FFT, orthogonal wavelet, biorthogonal wavelet, DCT, BWT, RLE, Huffman coding, Shannon-Fano coding, arithmetic coding, KLT, RGB, CMY, YUV, HSV, black body, Visible Human Project

Abstrakt

Účelem dizertační práce je předvedení nového přístupu k analýze objemových dat a zpracování dvojrozměrného obrazu. Metoda vizualizace objemových dat je založena na simulaci lidského vjemu při pozorování objemového objektu. Konkrétně jsou použity techniky lineárně perspektivního zobrazení, řešení viditelností pomocí ray-castingu a techniky kombinování barev. Proto je možno pozorovat vlastnosti objemového objektu přirozeně bez nutnosti znalosti těchto metod. Další část dizertační práce je věnována principům a aplikaci digitálních filtrů na obrazová data, kde jsou také použity techniky adaptivního filtrování jako např. konvoluce s adaptivním jádrem s konkrétním příkladem použití. Dále jsou prezentovány principy radix 2 FFT algoritmů a některé filtry, které je používají. Ke snížení celkové informační náročnosti obrazových dat jsou navrženy dva typy kompresních algoritmů. První algoritmus pro bezeztrátovou kompresi je je založen na Burrows-Wheelerově třídění bloků a adaptivním aritmetickým kódováním s konečnokontextovou predikcí. Druhý je zaměřen na ztrátovou kompresi obrazových dat, který je založen na progresivním DCT kódování. Také jsou prezentovány pricipy a příklady ortogonální a biortogonální waveletové transformace s aplikací při kompresi obrazu.

Klíčová slova:
objemový rendering, raycasting, konvoluce, FFT, ortogonální wavelety, biortogonální wavelety, DCT, BWT, RLE, Huffmanovo kódování, Shannon-Fanovo kódování, arithmetické kódování, KLT, RGB, CMY, YUV, HSV, absolutně černé těleso, Visible Human Project