

Charles University in Prague  
Faculty of Mathematics and Physics

## BACHELOR THESIS



Andrej Chovanec

## Digital Reconstructed Radiography

Department of Software and Computer Science Education

Thesis supervisor: RNDr. Josef Pelikán

Study program: Computer Science, General Computer Science

2009

I would like to thank my supervisor, RNDr. Josef Pelikán, who allowed me to work on this topic and advised me during the creation of both application and thesis. Moreover, I would like to thank MUDr. Martin Horák (Nemocnica na Homolce), who provided us with testing data and his expert opinions.

I declare that I have written my bachelor thesis independently and solely using cited sources. I agree with lending of the thesis and its publishing.

In Prague, 05/29/2009

Andrej Chovanec

# Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Motivation for digital reconstructed radiography . . . . .	9
1.2	Contribution of DRR . . . . .	10
1.3	Disadvantages of DRR . . . . .	11
<b>2</b>	<b>X-rays attenuation model</b>	<b>13</b>
2.1	Energy absorption . . . . .	13
2.2	Radiation scattering . . . . .	14
2.3	Beam hardening . . . . .	15
<b>3</b>	<b>Computed tomography</b>	<b>17</b>
3.1	Data acquisition . . . . .	17
3.2	Voxels . . . . .	18
3.3	Hounsfield units . . . . .	19
<b>4</b>	<b>Image reconstruction algorithm</b>	<b>20</b>
4.1	Input data . . . . .	20
4.2	X-ray apparatus representation . . . . .	22
4.3	Rays simulation . . . . .	23
4.4	Sampling values along the ray . . . . .	24
4.5	Single value evaluation . . . . .	27
4.6	Colour calculation . . . . .	29
4.7	Algorithm summary . . . . .	32
4.8	Tissue selection . . . . .	33
4.9	Edge enhancing . . . . .	33
4.10	Rotation . . . . .	36
4.10.1	Virtual trackball . . . . .	36
4.10.2	Adaptive resolution changing . . . . .	36
4.10.3	Bilinear interpolation . . . . .	37

4.11 Parallelization . . . . .	37
4.11.1 Raycasting parallelization . . . . .	39
4.11.2 Postprocessing parallelization . . . . .	40
4.11.3 Results of time improvement . . . . .	40
<b>5 Conclusion</b>	<b>41</b>
<b>Bibliography</b>	<b>43</b>
<b>A Contents of CD</b>	<b>45</b>
<b>B User guide</b>	<b>46</b>
B.1 Installation and launching . . . . .	46
B.2 Data loading . . . . .	46
B.3 Settings window . . . . .	48
B.3.1 Basic settings . . . . .	48
B.3.2 Advanced settings . . . . .	49
B.4 Rotation . . . . .	50
<b>C Implementation details</b>	<b>52</b>
C.1 Compilation . . . . .	52
C.2 Shooting and sampling the ray . . . . .	52
C.3 Threads . . . . .	54
C.4 Virtual trackball . . . . .	57
C.5 Marr filter . . . . .	59
<b>D Gallery</b>	<b>61</b>

# List of Figures

2.1	Exponential attenuation of x-ray beam . . . . .	14
3.1	Example of body discretization into voxels . . . . .	18
4.1	Transition from pixels representation to voxels representation	21
4.2	Perspective ray casting . . . . .	24
4.3	Sampling constant number of equidistant positions . . . . .	25
4.4	Sampling variable number of equidistant positions . . . . .	26
4.5	Sampling positions with 3D DDA algorithm . . . . .	27
4.6	Principle of interpolations . . . . .	28
4.7	X-ray image: pelvis, different interpolations . . . . .	30
4.8	X-ray image: trunk, different tissue selections . . . . .	34
4.9	X-ray image: skull, edge enhancing turned off and on . . . . .	35
4.10	X-ray image: skull model, lowered resolution . . . . .	38
B.1	Search window for loading datasets . . . . .	47
B.2	Series selector for selecting dataset . . . . .	47
B.3	Basic settings window . . . . .	48
B.4	Advanced settings window . . . . .	49
C.1	Two points $A$ and $B$ on the trackball . . . . .	58
D.1	Skull image with a noticable sinus . . . . .	61
D.2	Two different views at a steel rod in a hip joint . . . . .	62
D.3	Two different screens of the skull . . . . .	63

# List of Tables

3.1	CT numbers and related tissues . . . . .	19
4.1	Computational time of the algorithm using nearest-neighbour and trilinear interpolations . . . . .	29
4.2	Running times of the algorithm with different number of pro- cessors' cores . . . . .	40
C.1	Two kernels for approximating the Laplacian . . . . .	59
C.2	Discrete approximation of LoG operator for $\sigma = 1.4$ . . . . .	60

Název práce: Digitálne rekonštruovaná rádiografia  
Autor: Andrej Chovanec  
Katedra (ústav): Kabinet software a výuky informatiky  
Vedoucí bakalářské práce: RNDr. Josef Pelikán  
e-mail vedoucího: Josef.Pelikan@mff.cuni.cz

Abstrakt: Röntgenové vyšetrenie zohráva v medicínskej praxi dôležitú rolu, no popri všetkých výhodách, ktoré prináša, obsahuje aj isté obmedzujúce nedostatky. V tejto práci detailne popisujeme techniku, ktorá na základe vopred získaných CT dát dokáže zrekonštruovať digitálny röntgenový snímok tak, aby sa odstránilo čo najviac nevýhod tradičného snímkovania. Medzi hlavné vylepšenia nad klasickou rádiografiou patrí tvorba snímku z úplne ľubovlného pohľadu a interaktívne otáčanie snímku v reálnom čase. Počas rekonštrukcie sa kladie veľký dôraz na presné simulovanie fyzikálnych vlastností röntgenového žiarenia, s cieľom obdržať čo najkvalitnejší finálny obrázok. Nemalé úsilie je takisto vynaložené na dosiahnutie krátkeho výpočetného času, potrebného k vygenerovaniu jedného snímku. Za týmto účelom uvádzame model paralelizácie, ktorý rozložením práce niektorých komponentov na viacero jadier procesorov významne urýchľuje algoritmus.

Klíčová slova: röntgenové žiarenie, digitálna radiografia, počítačová tomografia, digitálna rekonštrukcia

Title: Digital reconstructed radiography

Author: Andrej Chovanec

Department: Department of Software and Computer Science Education

Supervisor: RNDr. Josef Pelikán

Supervisor's e-mail address: Josef.Pelikan@mff.cuni.cz

Abstract: X-ray examination is an important part of the medical treatment. Despite all the advantages it introduces, it brings some limitations as well. In the present work we describe a technique that from the acquired CT data reconstructs a digital x-ray image and removes some drawbacks of traditional x-ray screening. Among the most significant improvements over the classical radiography belong generation of the screen from very arbitrary angle and interactive rotation of the image in real-time. In order to obtain the most realistic final image, we put the emphasis on the accurate simulation of physical properties of the x-ray radiation. We also try to get as low computational time needed to gain one image as possible. For this purpose we present a parallelization model that decomposes the required work of some components into several processors' cores and thus noticeably decreases the running time of the algorithm.

Keywords: x-rays, digital radiography, computed tomography, digital reconstruction



# Chapter 1

## Introduction

### 1.1 Motivation for digital reconstructed radiography

In our everyday life people are often confronted with a lot of injuries, when it is necessary to transport person into the hospital and examine his/her health. Among these situations belong different traffic accidents, work injuries, collisions and many others. After the participants have been moved to the medical centre, they usually undergo lots of examinations. *Computed tomography*<sup>1</sup> and classical *x-ray screening*<sup>2</sup> belong to the routine ones.

X-ray screening is a method providing us possibility to see inner tissues of a human body. The result of the screening is a two-dimensional image making us able to locate fractures or damaged organs which normally cannot be seen by a naked eye. Having set up x-ray apparatus doctor can focus on a particular part of the body and in a few minutes he is given a detailed screen. On the other hand, x-ray examination has some potential drawbacks, some of them are:

- positions, which a patient can be screened in, are very limited. Because of the construction of the whole apparatus it is impossible to produce images of an arbitrary body part and under arbitrary angle of x-ray tube

---

<sup>1</sup>often abbreviated as CT

<sup>2</sup>also called röntgen (RTG) screening after one of its first investigators, Wilhelm Conrad Röntgen

- doctor often needs a detailed screen from a very exact angle. As long as the given screen does not meet this requirement, either the patient must undergo the whole procedure again (and that means to undergo x-ray dose too) or the doctor cannot give his best opinion
- very important issue about radiology<sup>3</sup> is radiation, which limits us in a number of eventually undergone tests. The more examinations taken, the higher risk of health problems.

Since 1970s, when digital imaging modalities such as computed tomography, ultrasound and nuclear medicine gained widespread acceptance, there has been made a lot of attempts to find new approaches how to avoid some of the problems of traditional radiography.[3] It is hard to imagine the way the current screening technology solves those problems and thus the new approaches are based on an entirely different idea. The main concept is that we will not get an x-ray screen directly from a screening device but we will *reconstruct it digitally from another acquired modality*. The most suitable form of input data is CT data and branch of science dealing with generating digital x-ray screens from CT is called **digital reconstructed radiography**<sup>4</sup>.

## 1.2 Contribution of DRR

Advantages of DRR can be divided into two main groups: ones which improve currently used methods and others presenting completely new possibilities in radio examination which were unfeasible with present technology.

The first group mostly concentrates on a better management and storage of x-ray films. A final result of a DRR is a two-dimensional *digital* image (digital reconstructed radiograph). An electronic form of the screen provides lots of enhancements over the films:

- x-ray film emulsion is composed of a form of gelatin and a silver halide, typically silver bromide.[11] These films have several limitations. They have a limited linear response to radiation, which means that it cannot tolerate a wide range in radiation exposure without risking saturation.

---

<sup>3</sup>radiology is a specialty of medicine that deals with the study and application of imaging technology like x-ray to diagnosing and treating disease[17]

<sup>4</sup>often abbreviated as DRR

The latitude limitations means some areas will be overexposed and some underexposed in the same film.[6]

- traditional image cannot be adjusted once taken. With DRR the image can be subsequently scaled, cut or rotated. Due to windowing<sup>5</sup> we are able to change brightness and contrast of the screen in any way to highlight its particular parts. However, some errors, such as positioning problems or patient movement reduce image quality regardless of the technology.
- finally, traditional radiography requires handling of film for viewing, archiving and transmission to others, which costs time. If a patient or another doctor wants to see screen remotely, the film must be sent via courier or scanned before electronic transmission.

The second group removes primarily problems related to potential positions the patient can be screened in:

- DRR allows us to choose arbitrary angle of the x-ray tube
- from one acquired CT dataset we can make as many different images as desirable
- if we want to focus on a very concrete part of a body, we can get x-ray tube as close as possible without danger of radiation dose
- we can select which types of tissue should be taken into consideration during image generation
- it is possible to interactively rotate x-ray tube, thus controlling the screening process itself is very comfortable

## 1.3 Disadvantages of DRR

On the other hand, there are areas where digital reconstructed radiographs can never exceed traditional films:

---

<sup>5</sup>windowing is the process of adjusting visualization of the calculated values. A typical display device can only resolve 256 shades of gray, some special medical displays can resolve up to 1024 shades of gray. By windowing these shades can be redistributed over a varying range of computed values[16]

- the most important one is an output resolution. While classical film's resolution is limited by molecules of the film emulsion, digital reconstructed image's resolution is much lower. Major cause of a low resolution is a speed of image reconstruction. The higher resolution, the higher computational time is required. Even if we wanted resolution to the exclusion of speed, the final quality would not change rapidly due to discrete input data.
- it is necessary to have recent CT data always when generating image. That means, the DRR cannot replace traditional radiography because exposure of CT is much higher than that of classical x-ray.
- the third drawback is more general and is related to the problems with storing data on the computer. It can be stolen, removed by hardware failure, etc.

In the following chapters we will first discuss some of the necessary theoretical topics before explaining the image reconstruction process itself. In chapter 2 we show a physical background of x-rays and their attenuation when traversing a matter, chapter 3 deals with principles of a computed tomography and finally in chapter 4 we describe step-by-step the whole process of digital x-ray image generation, which was used while implementing our own application.

# Chapter 2

## X-rays attenuation model

X-radiation (composed of x-rays) is a form of electromagnetic radiation. X-rays have a wavelength in the range of 10 to 0.01 nanometers, corresponding to frequencies in the range 30 petahertz to 30 exahertz and energies in the range 120 eV to 120 keV.[18] As x-rays traverse a matter the number of photons in a beam decreases due to interactions with the atoms of a material substance. Attenuation is caused primarily by two processes, absorption and scattering.[10]

### 2.1 Energy absorption

In absorption, the energy of the x-ray photon is completely transferred to the atoms of the material. Rate of beam absorption is expressed by a *linear attenuation coefficient*, which reflects the removal of x-ray photons from a beam. The higher the electron density, the more interaction of photons with the sample occurs. Assuming that x-rays are monoenergetic and the sample is homogeneous, the Beer's law gives us

$$I_{output} = I_{initial} \cdot e^{-\mu \Delta} \quad (2.1)$$

where  $I_{output}$  is an energy of the beam at the end of the sample,  $I_{initial}$  is an input energy,  $\mu$  is sample's linear attenuation coefficient and  $\Delta$  is sample's width.

For composite materials, the intensity is given by adding the individual contributions of each chemical element. Thus the resulting energy is:

$$I_{output} = I_{initial} \cdot e^{-\int \varphi(x) dx} \quad (2.2)$$

where  $\varphi(x)$  assigns attenuation coefficient to each position  $x$  on the ray. While working with discrete values, equation 2.2 can be rewritten as

$$I_{output} = I_{initial} \cdot e^{-\sum \mu_i \cdot \Delta_i} \quad (2.3)$$

where  $\mu_i$  is attenuation coefficient of the  $i$ -th element and  $\Delta_i$  is its width. This is illustrated in Fig. 2.1.

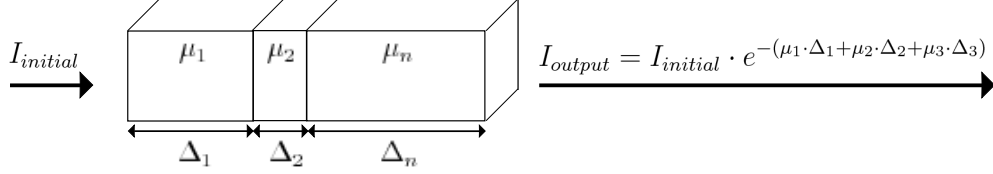


Figure 2.1: Exponential attenuation of x-ray beam

In x-ray imaging what is actually measured is not the final x-ray intensity, but the *blackening (the density)* of an optical film. This density depends on the type of film and is also specific for the x-ray energies used. The density  $d$  of the film is defined as[13]:

$$d = \log \frac{I_{initial}}{I_{output}} \quad (2.4)$$

Now if we replace  $I_{output}$  with the formula from 2.3, the blackening is given by

$$d = \sum \mu_i \cdot \Delta_i \quad (2.5)$$

## 2.2 Radiation scattering

Unlike absorption, when the energy of the x-ray photon is completely transferred to the atoms of the material, in the scattering the x-ray photon continues with a change in direction with or without a loss in energy. There are two types of scattering[9]:

**coherent scattering** is a scattering when the energy of the primary x-ray photon is first completely absorbed and then re-emitted by the electrons of a single atom. Because no net energy is absorbed by the atom, the re-emitted x-ray has the same energy as the original x-ray, however the direction of re-emission is totally arbitrary.

**Compton scattering** is a process when some of the photon energy is used to eject an electron from an incident atom and the photon is scattered with a reduced energy. The great majority of scattered x-rays in diagnostic x-ray imaging arise from this type of scattering.

Because the scattered radiation deviates from the straight line path between the x-ray focus and the image receptor, scattered radiation is a major source of image degradation in x-ray imaging techniques. The residual scatter reduces radiographic contrast in x-ray imaging and contributes to image intensity distortion in computed tomography.

Although the scattered x-ray photons are nearly isotropic in direction at diagnostic energies, the scattered x-ray detected in the image are primarily forward directed and thus have energies and angles of incidence near those of the primary x-rays and thus, these scattered x-rays cannot be completely removed by the use of antiscatter grids or energy filters.

## 2.3 Beam hardening

In the equations 2.1-2.3 we supposed that a particular volume element attenuates all x-rays in the same way independent of the angle or propagation path length. In fact, this assumption is not correct due to phenomenon called *beam hardening*. Beam hardening is a process referring to a polychromatic<sup>1</sup> beam when photons with higher attenuation coefficients are removed from the beam more rapidly. As the x-rays are always polychromatic, this nonuniform attenuation of different energies results in the preferential depletion of x-rays in energy ranges with higher attenuation coefficients. Thus, beam hardening is the process of selective removal of soft x-rays<sup>2</sup> from the x-ray beam. As these x-rays are removed, the beam becomes progressively harder or more penetrating.[8]

The amount of beam hardening depends on the initial x-ray spectrum as well as on the composition of the material or tissue traversed. However, for any fixed initial x-ray spectrum and tissue type, the process of beam hardening represents a monotonic increase in beam hardness as a function of tissue thickness traversed.[7]

Theoretically, emitted photons can reach an energy between  $E_{min} > 0eV$  and  $E_{max} = U_b \cdot e$ , where  $U_b$  is the maximum x-ray tube voltage and  $e$  the

---

<sup>1</sup>spectrum with multiple energies

<sup>2</sup>rays in energy ranges that are more easily attenuated

elementary charge<sup>3</sup>. If we denote  $W(E)$ <sup>4</sup> a weighting factor for each present energy, output energy for polychromatic beam is defined as[4]:

$$I_{poly} = I_{input} \cdot \int_{E_{min}}^{E_{max}} W(E) \cdot e^{-\int \varphi(x) dx} dE \quad (2.6)$$

The amount of needed correction of beam hardening equals to difference of 2.2 and 2.6. Some different approaches how compute  $I_{poly}$  can be found in [4].

---

<sup>3</sup> $e = 1.6021764610 \cdot 10^{-19}$  coulombs

<sup>4</sup> $W(E) = S(E) \cdot D(E)$ , where  $S(E)$  is a polychromatic spectrum and  $D(E)$  is detector's energy dependent efficiency[4]



# Chapter 3

## Computed tomography

A basic problem in imaging with x-rays is that a two-dimensional image is obtained of a three-dimensional object. This means that structures can overlap in the final image, even though they are completely separate in the object. This is particularly troublesome in medical diagnosis where there are many anatomic structures that can interfere with what the doctor is trying to see. This problem was solved with the introduction of a technique called computed tomography which takes advantage of movable x-ray tube and detector along the patient's body and generates one screen for each position of the pair. After acquiring and merging all data we get tridimensional model of examined body.

### 3.1 Data acquisition

A main principle of the computed tomography is similar to the one of the traditional x-ray imaging, when final image is generated on the base of differently attenuated x-rays. Unlike classical x-ray examination the x-ray tube and the detector are not static. They make two basic motions - the first one is transversal<sup>1</sup> (call it  $z$  direction) along the patient's body and the second one is rotational around the one particular  $z$  coordinate.

The pair x-ray tube - detector starts at a default position. The collimated<sup>2</sup> beam of x-rays is radiated and single attenuations of rays are de-

---

<sup>1</sup>there is also another type of acquisition, called tilted acquisition, used with some special examinations[1]

<sup>2</sup>collimation is the use of metal plates, slots, bars, etc., to confine and direct radiation (e.g. x-rays or gamma-rays) to a specific region

tected. This is called *single projection*. Then the tube and the detector rotate about e.g.  $1^\circ$  and make another projection until they reach e.g.  $180^\circ$  or  $360^\circ$ [1]. After all projections have been collected they are reconstructed into the one image - *CT slice* and the tube with detector move to a new position in a  $z$  direction. Number of obtained slices depends on the range of examined patient's body.

## 3.2 Voxels

In CT imaging the body is mapped as discrete, contiguous volume elements called *voxels* (Fig. 3.1). As an x-ray beam crosses the body it is attenuated by all voxels it traverses. At the end of the acquisition process each voxel is given two parameters:  $x, y, z$  coordinates in the body and linear attenuation coefficient.

Each voxel has three dimensions.  $x$  and  $y$  dimensions are determined by the pixel's area in the CT image and  $z$  dimension is determined by the slice thickness.

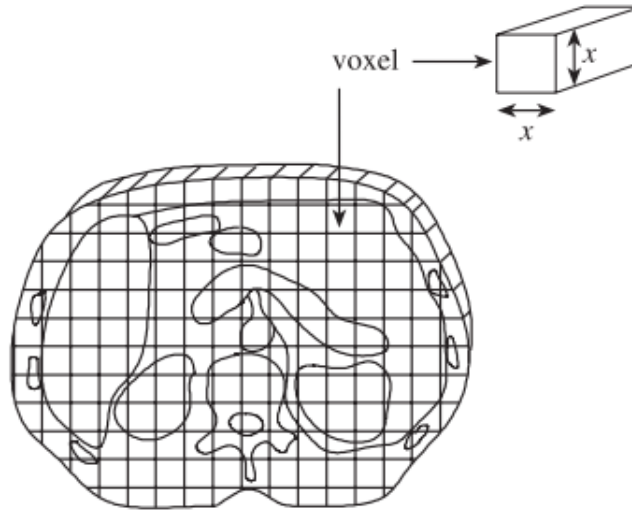


Figure 3.1: Example of body discretization into voxels. Greg Michael: X-ray computed tomography, 2001

### 3.3 Hounsfield units

We said one of the voxel's properties is its linear attenuation coefficient  $\mu$ . However, this physical unit is rather energy dependent<sup>3</sup> and direct visualization of  $\mu$  of the identical scene would differ with different devices which is impractical. For quantitative expression of image modality there were established new *Hounsfield units*.

Each voxel is given a *CT number* which depends on the original attenuation coefficient and on the attenuation coefficient of water characteristic for used energy. CT number is defined as[1]:

$$CT_{material} = K \cdot \frac{\mu_{material} - \mu_{water}}{\mu_{water}} \quad [HU] \quad (3.1)$$

where  $\mu_{material}$  and  $\mu_{water}$  are relevant attenuation coefficients and  $K$  is a contrast/scaling factor determined by a precision of measurement. Current CT devices reach contrast factor  $K = 1000$ . Values of CT numbers typical for particular tissues can be find in Fig. 3.1[15]:

Tissue type	CT number
Air	-1000
Fat	-80 to -40
H <sub>2</sub> O	0
Transudates/exudates	20 to 30
Soft tissue	40 to 80
Cancellous bone	150 to 300
Dense bone	300 to 2000

Table 3.1: CT numbers and related tissues

---

<sup>3</sup>e.g. for water for tube's voltages 60, 84 and 122 keV the coefficient is 0.206, 0.180 and 0.166[1]

# Chapter 4

## Image reconstruction algorithm

Aim of each DRR application is to produce high quality images resembling the real x-ray screens as much as possible. Moreover, the reconstruction process should be fast enough to provide efficient way of patient's examination and diagnosing.

The first goal is achieved by precise representation of x-ray imaging properties, like positioning of the x-ray tube and the film, film's resolution and following the real x-rays attenuation model. The second one is accomplished by selecting suitable programming techniques to reach the previous goal. Unfortunately, it is often not possible to gain these both requirements at once. We usually have to either choose between one of them or better find an appropriate compromise.

In this chapter we will describe reconstruction algorithm in detail, discuss problematic parts and also outline other possible solutions. In the individual sections we will use references and outputs from exclusively our own application, which was designed according to characterized algorithm.

### 4.1 Input data

In the very beginning of the algorithm we must choose which modality will be used as an input data. Because of the same physical background as traditional x-rays imaging the most suitable one is CT. As was said in chapter 3 CT data consists from multiple separate slices and each slice is formed by a matrix of pixels. Each dataset is characterized by a number of slices it contains and by a slice's resolution. While the usual resolution is 512x512 pixels the number of slices differs according to the range of scanned body. How-

ever, these two characteristics are not sufficient to obtain the real shape of a body. To get a true geometry of the data we need another two properties - distance between each two slices and size of the area represented by one pixel. With knowledge of all these parameters we get final representation as a 3D volume of voxels where its width and height is given by multiplying  $x$  and  $y$  resolution by a related size of the pixel area and depth is given by multiplying the number of slices by their distance. This is illustrated in the following Fig. 4.1:

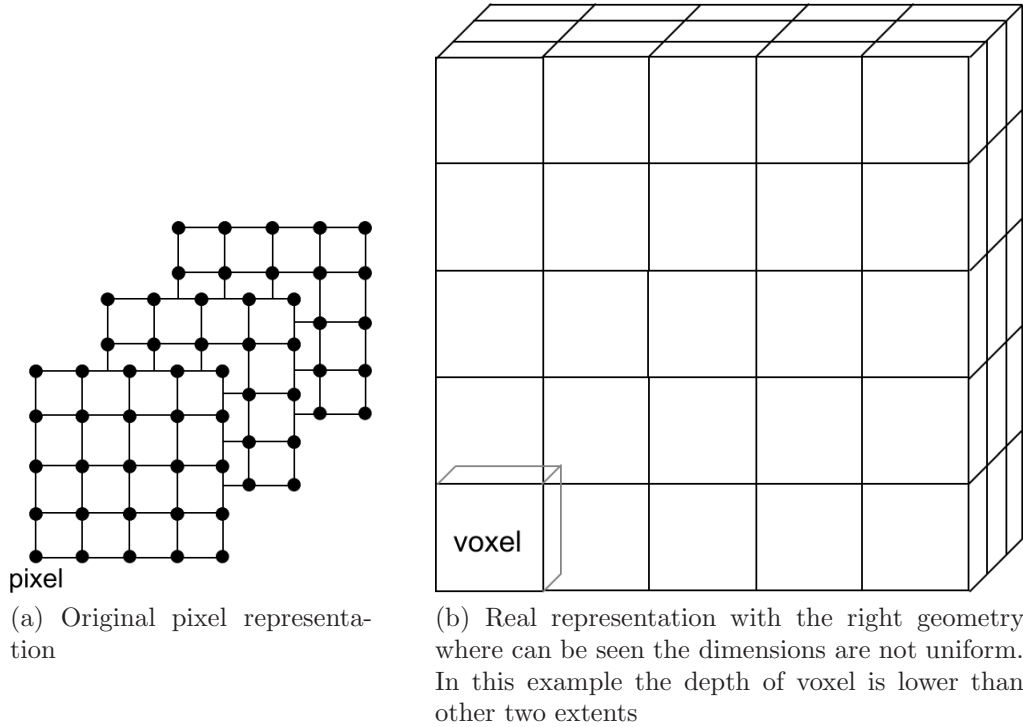


Figure 4.1: Transition from pixels representation to voxels representation

Except for preceding form of input data there is also another type depending on the settings of CT scanner while acquisition. In the previous example each slice represented one isolated section in the body. CT scanners can be switched to mode when they produce *overlapped* slices. Size of overlapping must be known to process data properly and each DRR application must be able to treat both types.

In Fig. 4.1b we showed representation where voxels are ordered in a regular grid. Due to a speed improvement of some algorithms (e.g. 3D DDA),

it may be sometimes profitable to choose another representation - so called *hierarchical spatial enumeration*[5]. The main idea of this method is to put together same neighbour voxels to form a new one „big” voxel. Then as we traverse through the data we can process the big voxel at once instead of processing the same voxels one by one. Since CT input data does not contain wide areas of voxels of the same colour, the enhancement would not be noticeable and thus we did not include this technique.

## 4.2 X-ray apparatus representation

The main components of the screening process are *an examined object (e.g. patient)*, *an x-ray tube* and *a film*. In order to achieve the most realistic results in a DRR application we should obey all their mutual relations while representing them.

### Examined object

Input data as described in a section 4.1 are stored in a 3D matrix and after they have been loaded they do not need any further attention.

### X-ray tube

X-ray tube and its orientation determines the view how the final reconstructed image is presented to the user. It means the tube behaves like a point from which the user is looking at an input object and that is why we will from now refer to the x-ray tube as a *camera*.

Camera has two primary properties. First of all the camera is defined by its three coordinates in a space. Unlike the real x-ray tube, the camera’s position is not limited and its coordinates may attain arbitrary values. However, the distance between object and camera should be short enough to obtain detailed image and on the other hand long enough to avoid *perspective distortion*<sup>1</sup> that would degrade realistic appearance which is important.

The second parameter is a direction in which the tube radiates the rays. Since it is possible to freely rotate the camera around the body, it must be

---

<sup>1</sup>perspective distortion is the inevitable misrepresentation of three-dimensional space when projected onto a two-dimensional surface. The closer the viewing point is to the object, the more obvious the phenomenon is.

able to set a new center of rotation easily as well to allow us focus on a particular part of the body and examine it from any direction.

## Film

As we said we would replace words x-ray tube and camera, we will also denote a screening film as a *projection plane*.

The first parameter of a projection plane is its position. The position is derived from the situation of the camera and must meet these requirements:

- the plane is perpendicular to the directional vector of the camera
- the plane is situated *behind* the inspected object
- the plane is as *tight* behind the object as *possible*. In a practice this property is always followed and significantly contributes to an accurate reproduction.

To satisfy all of these prerequisites with no error they are all computed automatically in our application without any help of the user.

The second attribute of the projection plane is a *resolution*. As was mentioned in the introduction of the thesis, this issue is a major drawback of DRR. The time needed to generate the image changes linearly with the resolution of the plane. To preserve the smoothness of the application the final resolution of the image is constantly set to 512x512 pixels. There were also tests with resolution 1024x1024 pixels but the speed of reconstruction was unsatisfactory.

## 4.3 Rays simulation

Very natural approach for simulating the radiation of x-rays is a *ray casting*. Ray casting is a common method used in a computer graphics serving for direct volume rendering. In this technique for each pixel of the projection plane there is generated ray crossing the input data. As a ray traverses the body it accumulates characteristics given by incident voxels. At the end of the path the collected value is written to the corresponding pixel.

There are two types of projections used in a ray casting: orthogonal and perspective projection. In the first one each generated ray is orthogonal to the projection plane. However, this is different from the nature of the real

x-ray imaging. That is why in the application is used *perspective* projection where all rays origin at the same point - camera - and direct to their pixels in the projection plane. This is illustrated in Fig. 4.2.

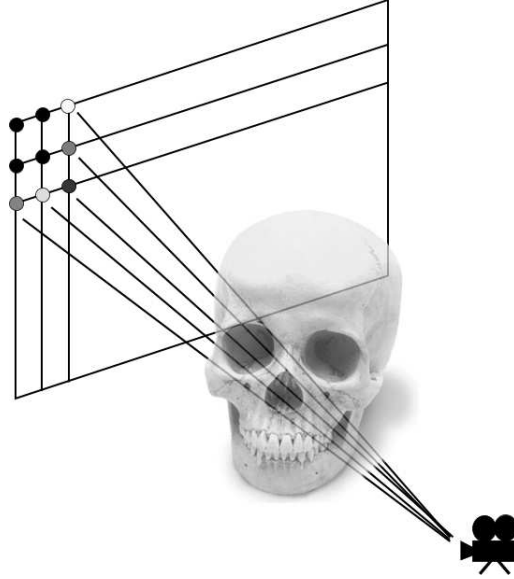


Figure 4.2: Perspective ray casting. Each pixel on the projection plane is given a colour accumulated along the ray. Skull image was borrowed from <http://www.fossilreplica.com/servlet/the-47/Human-Female-Skull-with/Detail>

## 4.4 Sampling values along the ray

Assumed the ray was generated, we need to pick concrete positions at which the value in the data will be evaluated. From all different approaches each one has its advantages and disadvantages. We always have to find a compromise between a good quality and a fast computation.

Basically there are three ways how to scan values: scan the *constant* number of equidistant positions for each ray, scan *variable* number of equidistant positions for each ray and finally 3D DDA algorithm.



### Constant number of equidistant positions

At the beginning of this method we select one constant value denoting the number of samples which will be taken along the each ray. Then the part of the ray which crosses the data is divided by this number into equally distanced positions (Fig. 4.3).

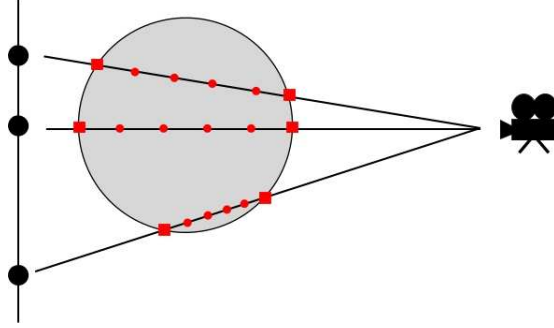


Figure 4.3: In this figure three rays were shot. For each one the entry and exit point was computed and then for chosen value 6 were calculated sampling positions (two red squares and four red circles).

#### *Advantages:*

- ability to choose the compromise between the speed and the quality. The more positions scanned, the higher quality and the lower speed and vice versa.

#### *Disadvantages:*

- if the distance between each two positions is too large, there is a risk that some „important” voxels will be skipped
- if the distance between each two positions is too low, the same value will be evaluated from the same voxel many times
- the major bottleneck is the number of the scanned positions which are independent of the ray's length. For the short rays there is uselessly large amount of positions which belong to a very few voxels.

### Variable number of equidistant positions

Likewise the previous method this one scans the positions in equal distances as well. However, the number of samples is not constant for all rays in advance, but is computed for each ray individually depending on its characteristics. As a one possible attribute is ray's length. This method can be seen in Fig. 4.4.

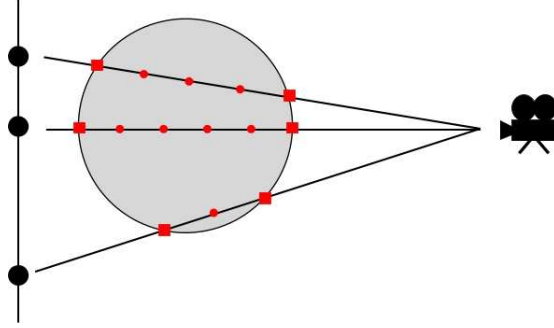


Figure 4.4: In this picture the number of sampled positions differs for each ray according to its length inside the data

#### *Advantages:*

- same as for the previous method
- in the short rays there are scanned only few positions which improves the computational time

#### *Disadvantages:*

- again, if we define a wrong relation between the length of a ray and the number of positions to be scanned, either too many voxel can be skipped or one voxel can be processed more times. Yet, there is still a significant improvement in comparison with the constant number of positions.

### 3D DDA algorithm

3D DDA algorithm[12] marks all positions along the ray where ray enters a new voxel. This guarantees all voxels are visited and each of them just once (Fig. 4.5).

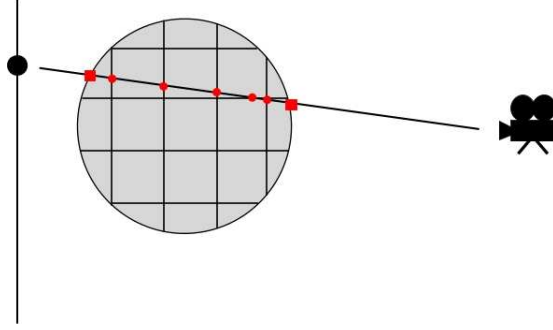


Figure 4.5: We can see that no two positions belong to the same voxel and no voxel is omitted.

*Advantages:*

- there is no chance any voxel along the path is skipped and thus no information from input data is lost

*Disadvantages:*

- a drawback of this method is its computational time

From these three methods it is important to choose that one which suits our actual requirements. In our application we chose the second method, because it is able to configure its properties, it is much faster than 3D DDA algorithm and it removes drawbacks of the first method. The number of positions is defined directly by the length of an intersection of the ray and data. It means, if the length of the intersection is 200, we will choose 200 equally distanced positions. Easy configuration is exploited in chapter 4.10 Rotation, where number of positions is set to 50% of the length to improve the speed.

## 4.5 Single value evaluation

Now suppose we have a particular position obtained by one from the sampling algorithms and we want to read the input on that position. If the position was exactly in the beginning of the voxel, we would return related

value. If the position is situated inside the voxel, the value must be *interpolated*<sup>2</sup> from the values around. There are more types of interpolation:

### Nearest-neighbour interpolation

This is the simplest interpolation when actual position is rounded to the nearest voxel and then a value is returned (Fig. 4.6a). Problem of this interpolation is it sometimes generates very evident and sharp changes of colour in the final image.

### Trilinear interpolation

Trilinear interpolation is a more sophisticated interpolation type where actual value is not calculated only from one voxel, but is calculated from all nearest surrounding voxels - it is their weighting (Fig. 4.6b). Trilinear interpolation produces smoother images, but at the cost of longer computational time.

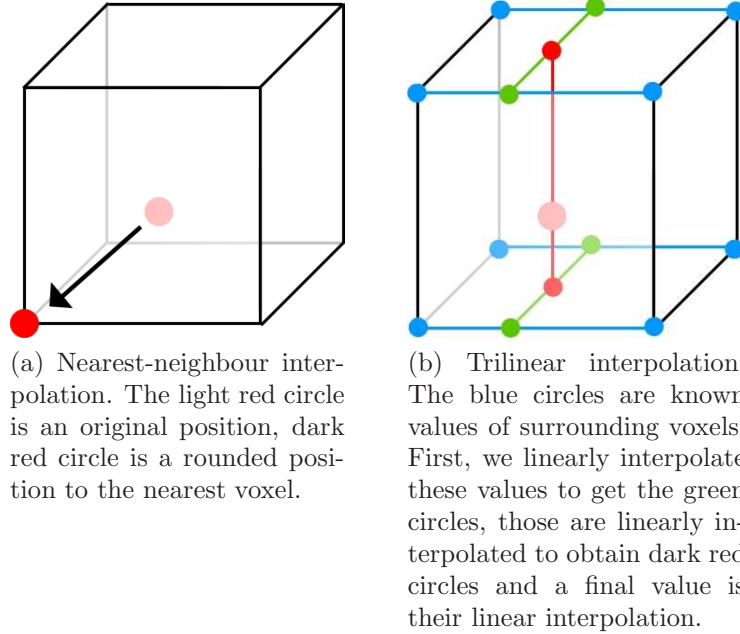


Figure 4.6: Principle of interpolations

---

<sup>2</sup>interpolation is a method of constructing new data points within the range of a discrete set of known data points

In most cases it is sufficient to use nearest-neighbour interpolation, which benefits from its speed. However, if the dataset had very long distances between each two slices, the quality of the result would be poor. In that case it is better to use trilinear interpolation. In Fig. 4.7 is shown how noticeably different the final image can be if we use distinct interpolations. It is on the user's choice in the settings window which method will be used.

Tab. 4.1 shows increase of the required computational time when using trilinear interpolation instead of nearest-neighbour interpolation.

Dataset [w×h×d]	Nearest-neighbour interp. [s]	Trilinear interp. [s]
512×512×305	4.11	5.42
512×512×49	10.50	12.98
512×512×73	13.73	17.03
512×512×91	18.06	22.36
512×512×400	5.53	7.40

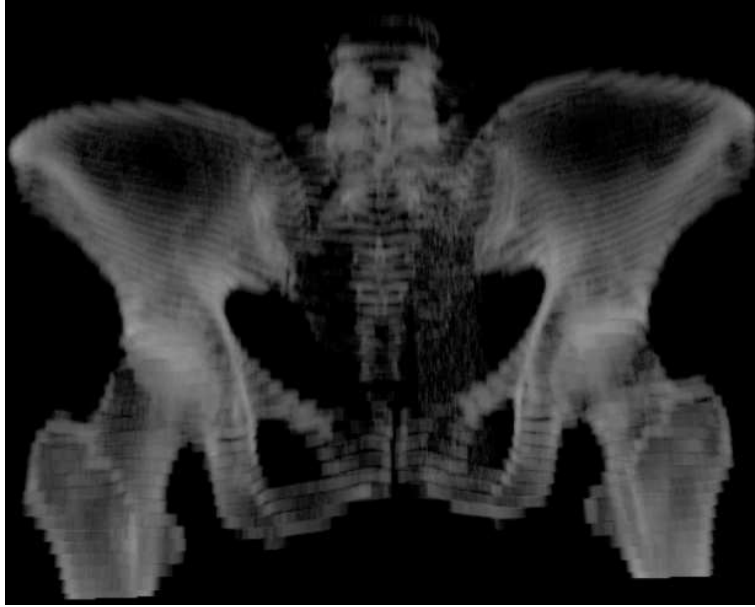
Table 4.1: Computational time of the algorithm using nearest-neighbour and trilinear interpolations

## 4.6 Colour calculation

The crucial part of the application is for a given ray calculate pixel's colour on the projection plane. Once the ray has been sampled and all relevant voxel values have been collected, we need to transform them into a colour that would be in a real x-ray picture. In order to get the most realistic digital image the accumulated value should be processed exactly according to the laws from chapter 2.

Imitating all characteristics of the x-rays is practically not possible and some of them have to be simplified:

1. First of all we ignore a polychromatism of the x-rays and work with them as they were monochromatic. This makes us able to avoid the equation 2.6 for an output intensity of the polychromatic beam which is very difficult to compute and use its simpler form in equations 2.1-2.3 instead. This simplification also removes a need to simulate a beam hardening, which is an inevitable phenomenon for a multienergetic spectrum.



(a) Nearest-neighbour interpolation used. The „steps” are very visible.



(b) Trilinear interpolation used. The image is obviously smoother.

Figure 4.7: Pelvis example with two different interpolation types. Input dataset had distance 5mm between each two slices.

2. We neglected the both types of scattering as well. We assume each ray is perfectly straight from its very beginning to its very end. If the scattering was included we would have to for each voxel generate new scattered rays which would markedly increase computational time of the algorithm.

With these assumptions the colour calculation is straightforward. As was said in chapter 2, the factor determining the final colour in the x-ray picture is its blackening. If we recall equations 2.4 and 2.5 it is obvious we even do not need the knowledge of the initial and output energy of the beam. According to equation 2.5 the only necessary information is linear attenuation coefficients along the ray and lengths of the intersections between the ray and the incident voxels. To get the all lengths of the intersections we must take care about it in a sampling algorithm (section 4.4). The linear attenuation coefficients can be obtained indirectly from the input CT data. In chapter 3 we said each CT slice is a matrix of CT numbers in Hounsfield units. If we know the attenuation coefficient of the water (which depends on the energies used), the attenuation coefficient of the particular voxel can be computed easily according to equation 3.1. With multiplying the attenuation coefficients with the related intersection lengths and summing it all together along the ray we get a number representing a colour.

However, this technique sometimes produced the images where the rear parts of the volume overlapped the parts in the front, which should be the most visible and the most consistent in the image. Therefore, we applied an improvement to our algorithm – before the linear attenuation coefficient is multiplied by the intersection length, it is nonlinearly transformed in a manner that the voxels closer to the camera are more important and contribute more to the cumulated colour. The only parameter of the transformation is the distance from the camera. Thus the final formula used to calculate the colour is

$$d = \sum e^{-\frac{i}{10}} \cdot \mu_i \cdot \Delta_i \quad (4.1)$$

Last required step is to assign a colour on the display to the number gained by a method described in the last paragraph. It is advantageous not to use an absolute relation between the number and the colour, but it is better to use colours relative to the range of the computed values. If we used an absolute relationship, the image would have very low contrast because most numbers belong to the narrow interval. If we use a relative relationship, the relatively small range of values is stretched over the wide

spectrum of grayscaled colours which improves the impression of the image. In order to get this effect we do a histogram equalization[2] on the image before showing it on the display.

## 4.7 Algorithm summary

Now if we put together all previous sections, the skeleton of the application is following:

1. Prepare input data
  - load the data from a medium
  - transform them to have the right geometry according to section 4.1
2. Initialize x-ray system
  - insert the transformed data into the coordinate system
  - set a default camera position and its view direction
  - initialize the projection plane, compute its position with respect to the camera, compute its distance to meet the requirements from section 4.2
3. For each pixel of the projection plane:
  - generate a ray with origin in a camera and direction to the actual pixel
  - find a start point and an end point of the intersection between the ray and input volume. If there is no intersection, write a default colour to the pixel
  - begin sampling the ray between the start and end point of the intersection and for each sample position:
    - compute the distance from the last position
    - evaluate a value at that position either by nearest-neighbour or trilinear interpolation
    - convert the CT number into the linear attenuation coefficient
    - multiply the distance and the coefficient and accumulate the result with the product from previous samples



- write the total accumulated value into the pixel
4. Postprocess final image<sup>3</sup>

## 4.8 Tissue selection

The principle of the algorithm we described allows us to selectively choose which tissues should be reflected in a final image. Tissue types are defined by the two boundaries of CT numbers determining the interval of „allowed” values. When we evaluate a value at a particular position on the ray we check whether the value belongs to the given interval or not. If does, the number is accumulated, otherwise we proceed directly to the next position.

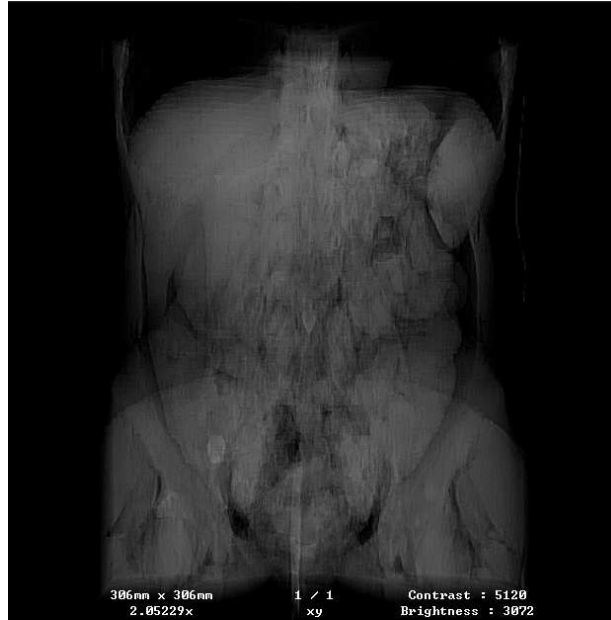
A goal of the tissue selection is not to provide some sophisticated techniques for such thing like organ segmentation. It is primarily meant to separate a bone tissue from all other tissues. In Fig. 4.8 we can see two reconstructed radiographs of the exactly same scene, but with different CT numbers interval. The interval is adjustable from the settings window of the application.

## 4.9 Edge enhancing

Sometimes it may be desirable to highlight certain features in the final picture. One of the postprocessing techniques is *edge enhancing*. There are several different algorithms used to highlight the edges. In our application we tested three of them and then chose the one with the best results. All of them were based on the convolution of the final image and suitable kernel. We experimented with Sobel operator, Laplace operator and Marr operator[2]. At last we decided to apply Marr operator which is advantageous over the previous two. Sobel operator generates too thick edges which may hide certain features and Laplace operator generates too many false edges which introduces noise to the picture. Moreover, the Marr operator is configurable in a sense of how sharp the edge should be to try to enhance it. Fig. 4.9 shows the original picture (Fig. 4.9a) and then picture enhanced with Marr operator (Fig. 4.9b). The second image has a little bit clearer edges at the cost of introducing some noise.

---

<sup>3</sup>postprocessing is not a necessary part of the algorithm and is placed in a separate section (4.9)

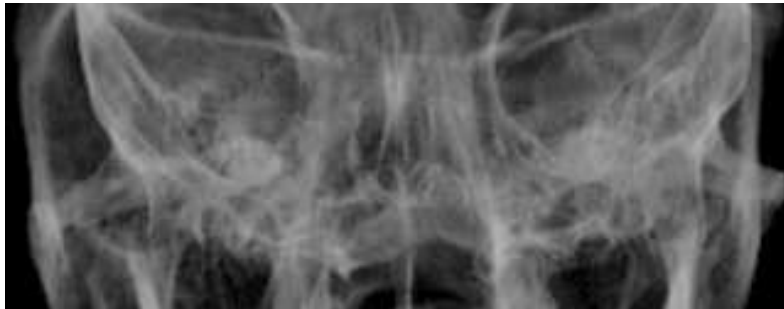


(a) Interval of CT numbers set to  $\langle 1000, 3500 \rangle$



(b) Interval of CT numbers set to  $\langle 1200, 3500 \rangle$

Figure 4.8: Trunk example with two different CT numbers intervals



(a) An original picture



(b) An enhanced picture

Figure 4.9: Two parts of the skull x-ray image, the second one with enhanced edges

## 4.10 Rotation

Although the DRR gives us possibility to generate digital screens from any angle, defining camera's position by three spatial coordinates is very uncomfortable. That is why we developed another approach for controlling the direction of the reconstruction. In our application we use a mouse to interactively rotate the x-ray tube which provides very easy and effective manipulation with the data.

### 4.10.1 Virtual trackball

To simulate the rotation we use a *virtual trackball*. Virtual trackball is a sphere situated in the centre of coordinate system. When we press a mouse button and move the mouse, the trackball begins to rotate. The rate of the rotation depends on the place where the button was pressed and on the direction of mouse movement. That means in fact we always want to rotate just the sphere. Now if we have a mathematical apparatus how to rotate the trackball (see implementation details C.4), calculation of the camera's position is easy. We imagine the camera is placed on our virtual sphere and as we rotate the sphere we rotate the camera according to the same rules. The same states for the projection plane. Each time the position of the camera changes we need to recalculate all parameters of the projection plane as well.

Default center of the trackball is in a point  $[0, 0, 0]$ . However, sometimes we may want to rotate around another point, e. g. we want to focus on a one bone and examine it from any angle. Then it is better to move the center of the trackball into this bone and start examination. We can use either mouse or a settings window to set the centre of rotation, both described in a user guide (B.3.2, B.4).

### 4.10.2 Adaptive resolution changing

To gain the experience of real-time rotating of the body it is necessary the response of the program was as fast as possible. Unfortunately, the computational time of the algorithm is not satisfactory. To preserve the smoothness of transition between two subsequent pictures we developed *adaptive resolution changing*. As we start to rotate the body, the resolution is automatically switched into lower values. When the mouse button is released, the image is

recalculated into the full quality. If we want to rotate the image again while the high quality recalculation has not finished, it is immediately stopped and the resolution is lowered and prepared for prompt response. As a good compromise between the speed and the quality on our dual core processor we set the low quality resolution to be 16 times lower than the full quality resolution by default. This value can be whenever changed through the settings window of the application.

### 4.10.3 Bilinear interpolation

It is possible to get the good final results despite the low resolution. The solution is to bilinearly interpolate the low resolution image. Although this interpolation is performed each time the picture is rotated, it is fast enough not to break the smoothness of rotation.

If the resolution is lowered 16 times, it means we generate rays only for each fourth pixel of the projection plane in vertical and horizontal direction. The others pixels are then interpolated from the surrounding four calculated pixels. In Fig. 4.10 we show the original image (Fig. 4.10a) and then bilinearly interpolated picture with low resolution (Fig. 4.10b). As we can see, the quality is very satisfactory even the resolution is 16 times lower.

## 4.11 Parallelization

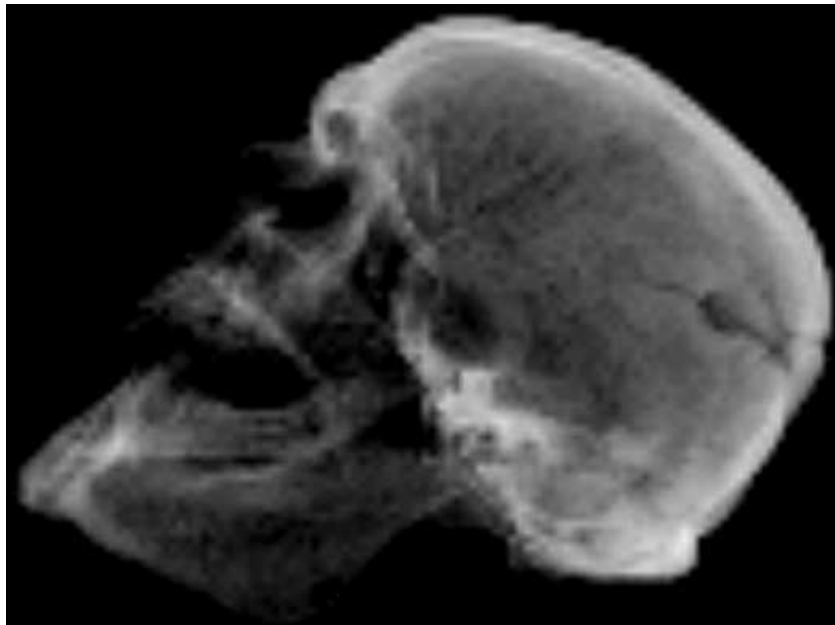
Besides degrading the output resolution there are also other possibilities how to accelerate the reconstruction and significantly decrease its running time. Among feasible ways belong:

- upgrading the processor
- improve used algorithms
- use parallelization

As in a medical environment it is not possible to whenever change the hardware because it would be costly and we tried to implement each algorithm as efficiently as possible, the only reasonable solution is a parallelization. Parallelization requires more threads to be run at once and all multicore processors meet this requirement. In our algorithm there are several places that may run simultaneously.



(a) An original picture in a full quality



(b) A picture with 16 times lower resolution, but bilinearly interpolated.

Figure 4.10: X-ray images of the skull model. The second one has very good quality despite the low resolution.

### 4.11.1 Raycasting parallelization

Parallelization of raycasting consists in division the projection plane into certain number of mutually independent areas and then performing raycasting on each of them separately. The way how the projection plane is divided is very important and there have to be more criteria taken into consideration:

1. number of the independent areas

It is wrong to assume that it is sufficient to divide the projection plane into the number of parts equal to the number of processors' cores. There may be a case (and in practice this is very often) that some part of the volume is processed longer than the other parts. This leads to the situation that some parts of the projection plane have already been processed and their cores *uselessly* wait until the overextended cores will finish their parts. Therefore we chose another approach. The projection plane is divided into many smaller parts whose number is much higher than the number of processors' cores. Then each free core is given one small part and as soon as the core is finished it is given a new unprocessed part. This implies that no core is idle while the other cores would work.

2. size of the areas

To select the most appropriate size of the areas we have to find a compromise between their size and their amount. The too large areas introduce problems mentioned in the previous paragraph. On the other hand, the too small areas introduce overhead related to the management of the threads. A good compromise is to choose number of pixels between 256 and 1024 pixels.

3. shape of the areas

At the first sight this criterion may seem unimportant. However, the shape of the areas determines how often we will *miss the cache*. If we divided the projection plane into the whole rows, there would certainly be more cache misses than if we took square areas.

According to these three points we decided to divide the projection plane into square areas, where each area is 32 pixels wide.

### 4.11.2 Postprocessing parallelization

The postprocessing technique that is parallelized is an interpolation. Bilinear interpolation is used during image rotation, as described in Sec. 4.10.3. As there are always required only four surrounding pixels for interpolating one pixel, the image may be easily divided into the several parts that are processed simultaneously. Unlike the raycasting parallelization, the image is divided into the predefined number of parts equal to number of cores, because all parts are computationally identical and there is no risk any core would stay idle.

### 4.11.3 Results of time improvement

In the Tab. 4.2 we present running time of the algorithm depending on the number of processor's cores.

	Single core [s]	Dual core [s]	Quad core [s]
1.	7.477	3.936	1.141
2.	7.447	3.939	1.156
3.	7.437	4.126	1.343
4.	11.516	5.975	
5.	22.541	11.946	
6.	4.006	2.141	

Table 4.2: Running times of the algorithm with different number of processors' cores. Single and dual core times were computed by AMD Turion 64 X2 TL-52, 1.61 GHz, 1024 MB RAM. Quad core results were computed by Intel Core 2 Quad Q6600, 2.40 GHz, 8192 MB RAM.



# Chapter 5

## Conclusion

In this work we presented a method for generating digital reconstructed radiograms and then according to described algorithm we implemented our own DRR application. In the first chapter we outlined the main objectives that every DRR program should accomplish – produce high quality digital x-ray screens and do it as fast and as comfortably as possible. We succeeded in fulfilling these particular goals:

1. **The final images were examined and verified by the expert specialized in radiology, MUDr. Martin Horák, who confirmed that quality of the results is good enough to help during a medical treatment.**

Our success was primarily achieved by the emphasis we put on the accurate simulation of the real screening process. We began with an exact representation of all components participating in screening process. Then we analyzed physical properties of x-rays radiation, mainly attenuation model, and implemented the formulas into the application. Although we ignored some phenomena of x-rays, like photons scattering and beam hardening, it did not noticeably affected the result at all.

2. To make the manipulation with the program comfortable, we developed real-time rotation of the input data. User is able to arbitrarily choose the position of the virtual x-ray tube by the mouse and thus make the examination effective. Without possibility to interactively manipulate the data, user would have to set x-ray tube's position as a

three coordinates in a 3D space. This approach is very clumsy because it is not intuitive what the coordinates should be like if we want to adjust the actual view direction.

3. We also managed to avoid the problems when the bone tissue is covered with another tissue that is unimportant in the particular situation. The method we used was a selective processing of voxels, when the voxels with value outside the selected CT interval were omitted.

Despite the satisfying results there is left a lot of areas in which the work is imperfect and could be improved.

1. There is always a scope for improving the quality of the final images. If we compare a real x-ray screen and a screen produced by our application, we can see the satisfactory resemblance, but certain difference is evident. Better results could be achieved by removing some simplifications (introduced in Sec. 4.6 *Colour calculation*) of x-rays properties.
2. In the current program state, any computations are performed on CPU. If we used GPU instead, the computational time would rapidly decrease.
3. Sometimes it is desirable to highlight a specific tissue type in the input volume. Our application lacks any means to adjust the transfer function.
4. If we move the centre of rotation and then rotate the x-ray tube, it is difficult to find out mutual position of the body and tube. It would be beneficial if we placed a small picture into the settings box showing centre of coordinate system, volume's boundaries, center of rotation and tube's position at once.

# Bibliography

- [1] Drastich A.: *Tomografické zobrazovací systémy*, Brno, 2004
- [2] Gonzales C. R., Woods E. R.: *Digital image processing, Third edition*, 2008
- [3] Kodak: *Introduction to digital radiography. The role of digital radiography in medical imaging*, 2000
- [4] Krumm M., Kasperl S., Franz M.: *Referenceless Beam Hardening Correction in 3D Computed Tomography Images of Multi-Material Objects*, China, 2008
- [5] Levoy M.: *Efficient Ray Tracing of Volume Data*, ACM Transactions on Graphics, 1990
- [6] Mattoon S. J., Smith C.: *Breakthroughs in Radiography: Computed Radiography*, 2004
- [7] Medcyclopaedia: *Beam hardening*,  
[http://www.medcyclopaedia.com/library/topics/volume\\_i/b/beam\\_hardening.aspx](http://www.medcyclopaedia.com/library/topics/volume_i/b/beam_hardening.aspx)
- [8] Medcyclopaedia: *Polychromatic x-ray beam*,  
[http://www.medcyclopaedia.com/library/topics/volume\\_i/p/polychromatic\\_x\\_ray\\_beam.aspx](http://www.medcyclopaedia.com/library/topics/volume_i/p/polychromatic_x_ray_beam.aspx)
- [9] Medcyclopaedia: *Scattered radiation*,  
[http://www.medcyclopaedia.com/library/topics/volume\\_i/s/scattered\\_radiation.aspx](http://www.medcyclopaedia.com/library/topics/volume_i/s/scattered_radiation.aspx)
- [10] Medcyclopaedia: *X-ray attenuation*,  
[http://www.medcyclopaedia.com/library/topics/volume\\_i/x/x\\_ray\\_attenuation.aspx](http://www.medcyclopaedia.com/library/topics/volume_i/x/x_ray_attenuation.aspx)

- [11] Medcyclopaedia: *X-ray film*,  
[http://www.medcyclopaedia.com/library/topics/volume.i/x/x\\_ray\\_film.aspx](http://www.medcyclopaedia.com/library/topics/volume.i/x/x_ray_film.aspx)
- [12] Pelikán J.: *Urýchlovací techniky (pro Ray-tracing)*, KSVI MFF UK Praha, 1996-2001
- [13] Schram R. P. C.: *X-ray attenuation: Application of X-ray imaging for density analysis*, Petten, 2001
- [14] Smith W. S.: *Digital Signal Processing: A Practical Guide for Engineers and Scientists*, 2002
- [15] Yang C., Guiney M., Hughes P., Leung S., Hoe Liew K., Matar J, Quong G.: *Use of digitally reconstructed radiographs in radiotherapy treatment planning and verification*, 2000
- [16] Wikipedia, The free encyclopedia: *Computed tomography, windowing*,  
[http://en.wikipedia.org/wiki/Computed\\_tomography#Windowing](http://en.wikipedia.org/wiki/Computed_tomography#Windowing)
- [17] Wikipedia, The free encyclopedia: *Radiology*,  
<http://en.wikipedia.org/wiki/Radiology>
- [18] Wikipedia, The free encyclopedia: *X-ray*,  
<http://en.wikipedia.org/wiki/X-ray>

# Appendix A

## Contents of CD

On the accompanied CD we can find:

- `/Datasets` – input data for testing purposes
- `/Documents`
  - `/Internal documentation` – *doxygen* generated documentation of the application
  - `/MedV4D` – user graphical interface documentation of the MedV4D framework
  - `/Thesis` – source files of this document in  $\text{\LaTeX}$  format and compiled *pdf* version
- `/Executables` – *exe* files of the application with all necessary files
- `/Screenshots` – contains outputs of the application
- `/Source codes` – complete source codes with Microsoft Visual Studio 2008 project files
- `contact.txt` – contact information in case of any opacities and questions


# Appendix B

## User guide

### B.1 Installation and launching

There is no need for any special installation of the application, it is just sufficient to copy files *CT2Xray.exe* and *config.cfg* into the same folder and then launch *CT2Xray.exe*. These two files can be found on the provided CD.

### B.2 Data loading

To load the input data we must either choose *File – Search* from the menu at the top of the screen or choose an  icon from the top of the screen. Then we are showed a Search window (Fig. B.1) where we can select location of the dataset. First we must choose what medium is the data on. We select *DICOMDIR* tab (mark ❶). Subsequently we find the folder with the data and press button *Search* (mark ❷). Then in the area at the bottom of the window (mark ❸) we are given all found datasets. We choose the one and double click it. Depending on the dataset, either the loaded data are showed or the next window will appear (Fig. B.2). In this window we can choose a concrete data if the dataset has more of them.

As we loaded our data, they are showed on the screen. How to control the whole graphical user interface of the application, please consult chapter 8 of the MedV4D documentation that can be found on the CD.

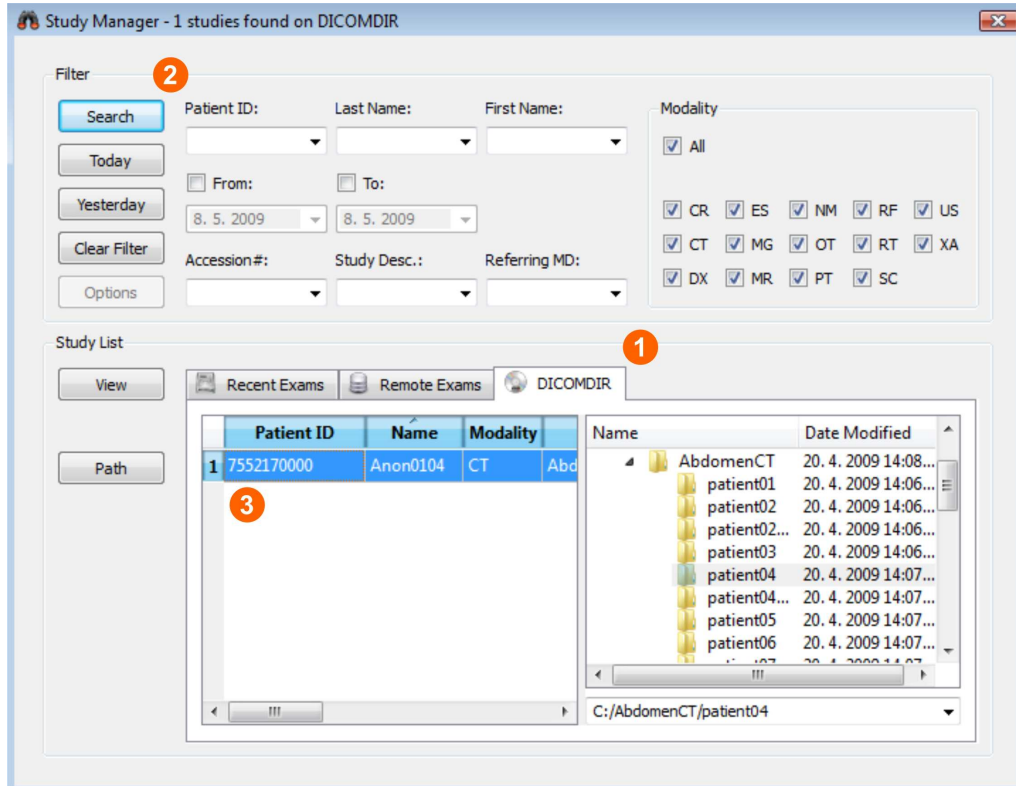


Figure B.1: Search window for loading datasets

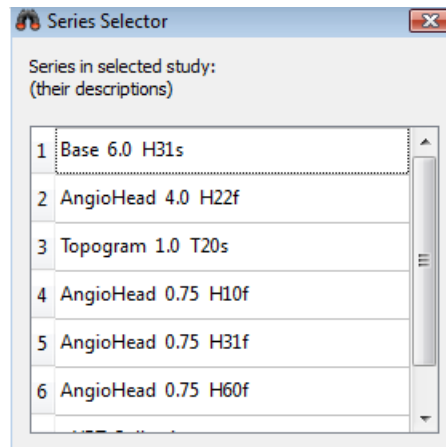


Figure B.2: It is possible to have more different datasets in a one dataset. This window allows us to choose the concrete one.

## B.3 Settings window

Behavior of the application can be adjusted in several ways through the *Settings window*. The settings are divided into two groups - basic settings and advanced settings. At the bottom of the window there is *Start filter* button, which executes the reconstruction process.

### B.3.1 Basic settings

Window with the basic settings (Fig. B.3) contains because of simplicity only those parameters that are used the most often.

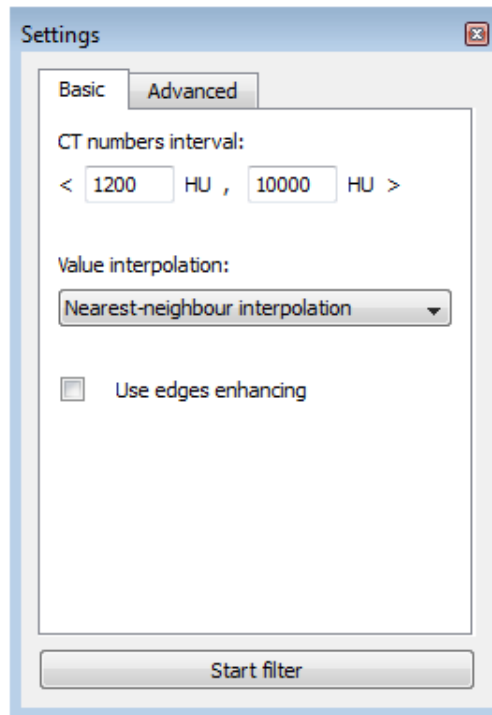


Figure B.3: Basic settings window

#### CT numbers interval

In section 4.8 we spoke about tissue selection. These two fields specify the interval of CT numbers which will be taken into consideration during reconstruction process.



### Value interpolation

This field defines the type of interpolation used in a reconstruction algorithm and determines the quality of the final image in general. The nearest-neighbour interpolation gives faster but poorer quality results and on the other hand the trilinear interpolation gives slower but higher quality results.

### Use edges enhancing

If checked, the reconstructed picture will be subsequently enhanced with sharper edges.

## B.3.2 Advanced settings

Advanced settings (Fig. B.4) are used to setup parameters that are not changed very often.

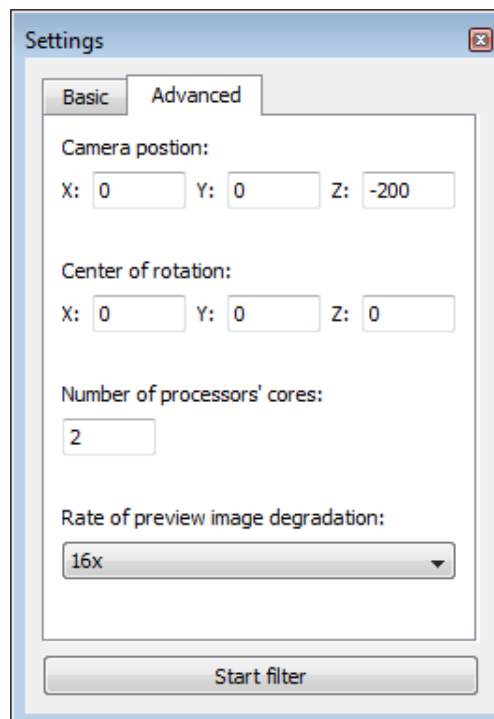


Figure B.4: Advanced settings window

### **Camera position**

These three fields show the actual coordinates of the camera in a virtual 3D space. By setting these values to concrete values and executing the filter we can get the camera exactly where we want.

### **Center of rotation**

Center of rotation defines the direction in which the camera is looking and defines a point around which camera rotates when we use a mouse to interactively select camera view.

### **Number of processors' cores**

Number of cores determines the rate of parallelization that will be used. The higher the number is, the more simultaneously parts of the application will be running. If we have a single core processor, we set the field to 1, if we have a dual core processor, set the value to 2 and so on.

### **Rate of image preview degradation**

Section 4.10 dealt with the need of lowering the resolution during interactive rotation. This field allows to choose the particular factor how much the resolution is lowered. The available options are  $1\times$ ,  $4\times$  and  $16\times$ . The first option will not degenerate the resolution at all, the second and the third options will lower it relevant times.

## **B.4 Rotation**

It is possible to use a mouse to control the reconstruction process. By the mouse we can arbitrary rotate the data, zoom the x-ray tube and change centre of rotation.

To rotate the data around the center of rotation we use a left mouse button. We start the rotation by pressing the button, then we move the mouse in any direction and stop rotating by releasing the button. As far as the button was released, the image is recalculated into full quality.

In the previous paragraph we said the rotation is performed around *a center of rotation*. Sometimes it is desirable to move the center of rotation and rotate around a different point. To move the center, we press the right

mouse button and move the mouse. To stop moving the center we release the button.

It is also possible to zoom in and zoom out the camera. There are two types of zooming. To zoom in larger steps we use scrolling by the roller. However, if we need to zoom in a very fine steps, we press the roller and move the mouse up and down, depending whether we want to zoom in or zoom out.

# Appendix C

## Implementation details

### C.1 Compilation

Our DRR application is based on the *MedV4D framework*. The goal of the MedV4D project is the development of a complex software for medical imaging data analysis and processing. Before we compile our project it is necessary to have prepared the libraries of the framework, concretely revision 1101. A guide how to compile MedV4D can be found at <http://cgg.ms.cuni.cz/trac/medv4d/wiki/HowtoDevel>.

To successfully compile the project we need to know location of the framework's libraries and header files. If we use a Microsoft Visual Studio, it is assumed all libraries are placed in a folder *MedV4D/trunk/src/lib* that is on the same place as is the main folder of the application. The header files are assumed to be in a folder *MedV4D/trunk/src/include*.

Compilation also supposes we have installed *boost library* from version 1.36.X.

### C.2 Shooting and sampling the ray

A class wrapping shooting and sampling rays is called *RayTracer*. The method that starts raycasting is `void start(int x, int y, int width, int height)`. The first two parameters define the starting position in the projection plane that is going to be processed. The next two parameters define area's width and height. Determining the particular part of the projection plane is used further in parallelization.

Rays generation is provided by a class `PerspectiveRayGenerator`. The class has three methods: `void init()`, `bool hasRay()` and `void nextRay(Ray & ray)`. Their names are self-explanatory. The part of the code generating the rays is following:

---

**Algorithm 1** Generating the rays

---

```

1: raysGenerator.init(startX, startY, width, height);

2: while raysGenerator.hasNextRay() do
3:   raysGenerator.nextRay(actRay);
4:   further steps
5: end while

```

---

Now suppose we have one ray from Alg. 1. The next step is to sample it. To do this we use a class `RegularPercentageRayStepper`. This class provides methods `void setRayStartEndPosition(rayStart, rayEnd)`, `bool hasNextPosition()` and `ValueType nextPosition()`. If we continue from the line 4 of the Alg. 1, the code is following:

---

**Algorithm 2** Sampling the ray

---

```

1: find an intersection of the ray and input data (rayStart, rayEnd);

2: rayStepper.setRayStartEndPosition(rayStart, rayEnd);
3: while rayStepper.hasNextPosition() do
4:   actualPosition  $\leftarrow$  rayStepper.nextPosition();
5:   further steps
6: end while

```

---

The last step is to evaluate the value at the given position and then accumulate the colour. To find out the concrete value in the input there are classes `NoInterpolationValueEvaluator` and `TriLerpValueEvaluator`. These static classes have only one method `ValueType getValue(actualPosition)`. The first of the two uses a nearest-neighbour interpolation, the second one uses a trilinear interpolation. To accumulate the value we use a `RTGAccumulatorExpoAtt` class. It has three methods – `void flush()`, `void addValue(const ValueType & value)` and `ValueType getResult()`. The first method resets the accumulator, the second one adds a new value and

the last one returns an accumulated result. Again, if we continue from the line 5 of the Alg. 2, the algorithm is:

---

**Algorithm 3** Evaluating and accumulating the colour

---

```

1: if user chose a nearest-neighbour interpolation then
2:    $value \leftarrow NoInterpolatValueEvaluator :: getValue(actualPosition);$ 
3: else
4:    $value \leftarrow TriLerpValueEvaluator :: getValue(actualPosition);$ 
5: end if

6:  $accumulator.addValue(value);$ 

```

---

## C.3 Threads

Parallelization is based on the execution of some application's parts in the individual threads. The application consists of these threads:

- main loop
- GUI events handler
- bilinear interpolator
- low and high quality raytracers<sup>1</sup>

To implement threads we use a *boost* library. A class invoking a new thread is `boost::thread`. Synchronization among the threads is ensured by `boost::mutex` and `boost::condition_variable` classes.

### Main loop thread

Main thread is always started at the beginning of the filter and takes care of correct objects initialization and launching other executive threads.

---

<sup>1</sup>from now and further we do not distinct the words raycasting and raytracing

## GUI events handler

Events handler is a thread running on the background and its aim is to collect and process all mouse and keyboard events. Class responsible for events is called `MouseHandler`. Interface of this class provides functions `mouseDoubleClickEvent()`, `mouseMoveEvent()`, `mousePressEvent()`, `mouseReleaseEvent()`, `wheelEvent()`, `keyPressEvent()` and `keyReleaseEvent()`.

## Bilinear interpolator

This thread is represented by class `boost::thread_group`. First we divide the projection plane into the known number of areas and then we interpolate each of them in a separate thread.

## Low and high quality raytracers

This is a set of threads performing the reconstruction process itself. Number of the simultaneously running threads depends on the user's settings, basically is derived from the number of processors' cores.

In section 4.11.1 we described how the parallelization of the raycasting is performed. First of all it is necessary to divide the projection plane into the small pieces and then assign them gradually to separate raytracers. A class handling the distribution of the pieces is called `ChunksManager`. `ChunksManager` is initialized with four parameters – width and height of the projection plane and width and height of each chunk (single piece of projection plane). The pseudo code for managing the chunks is following:

---

**Algorithm 4** Management of the chunks

---

```
1: chunksManager.reset();  
2: while chunksManager.hasChunk() do  
3:   chunk  $\leftarrow$  chunksManager.nextChunk();  
4:   startRayTracer(chunk.x, chunk.y, chunk.width, chunk.height);  
5: end while
```

---

It is important to always keep the same number of running raytracers at once equal to the number of disponsible processors' cores. As we can see in the Alg. 4, the chunks are being assigned one by one. To take care of the right amount of threads there is a class `RayTracersPool`. This class has

two main data members, `std::vector<ParallelRaytracer> raytracers` and `std::deque<int> freeRaytracers`. `raytracers` is an array of both free and employed raytracers. Each raytracer is given a unique *id* that identifies it and that is used as a key for queue `freeRaytracers`. This queue determines which raytracers are working and which are not. As far as there is a request to start a new raytracer, `RayTracersPool` checks the `freeRaytracers` at first and finds out whether it is possible to start a new thread. If `freeRaytracers` is empty, it means no raytracer is available and the pool must suspend execution of new threads until any raytracer finishes its work. Otherwise, the pool pops the front of the `freeRaytracers` and starts raytracer with that *id*. Alg. 5 sketches described steps:

---

**Algorithm 5** Execution of the new raytracers

---

```

1: while freeRaytracers is empty do
2:   sleep();
3: end while

4: while freeRaytracers has an item do
5:   id  $\leftarrow$  freeRaytracers.front();
6:   startThread(raytracers[id]);
7:   remove a front from freeRaytracers;
8: end while

```

---

Lines 1 – 3 show a loop waiting for any raytracer to finish. In practice, we do not use `sleep()` function, because it would stop the entire code. Instead we use a `boost::condition_variable` that can *block* itself and after notifying from another thread is able to continue. Usually it is a raytracer who calls the `notify()` function when finishes its work and pool immediately leaves the first *while loop*. The second loop (lines 4 – 8) takes *ids* of all free raytracers one by one and executes a new thread for each of them.

Often there is a place in the code when we need to wait until all raytracers finishe their work. `RayTracersPool` provides method `stopAndWaitAll()` (Alg. 6) that sends a signal to each raytracer to stop and then block itself until each raytracer definitely returns from a thread function.



---

**Algorithm 6** Stopping and waiting all raytracers to finish

---

```
1: for  $i = 1$  to  $maxNumberOfThreads$  do  
2:    $raytracers[i].stop()$ ;  
3: end for  
  
4: while  $freeRaytracers$  is not full do  
5:    $sleep()$ ;  
6: end while
```

---

Similar to line 2 of the algorithm 5, neither here we use a `sleep()` function, but we use a `boost::condition_variable` again.

## C.4 Virtual trackball

To control the virtual trackball we use a mouse moving in a 2D window. As the trackball is three dimensional we need to find a relation between 2D coordinates in a window and 3D coordinates on the trackball. It is good to choose the trackball with *unit radius*. Then the mapping algorithm is following:

1. Scale the  $x$  and  $y$  coordinates of the mouse into the interval  $[-1, 1]$ . New coordinates are defined as

$$scaledX = \frac{2 \cdot x}{width_{window}} - 1 \quad scaledY = 1 - \frac{2 \cdot y}{height_{window}} \quad (C.1)$$

This gives us the first two coordinates on the 3D trackball.

2. The third coordinate is computed from the formula

$$x^2 + y^2 + z^2 = r^2 \quad (C.2)$$

If we use a unit radius, the  $z$  coordinate is given by

$$z = \sqrt{1 - x^2 - y^2} \quad (C.3)$$

If the expression in a square root is negative, we set the coordinates

as

$$\begin{aligned}x &= \frac{x}{\sqrt{x^2 + y^2}} \\y &= \frac{y}{\sqrt{x^2 + y^2}} \\z &= 0\end{aligned}\tag{C.4}$$

3. The mapped point to the trackball has coordinates  $(x, y, z)$ .

Now when we have a procedure that maps a 2D point to a 3D point, we are able to compute the rotation matrix. When the user clicks on a point within our window, we will calculate corresponding point on the ball (call it  $A$ ). Then, as long as the mouse is moved with the button down we will compute the point on the ball corresponding to the pixel at which the mouse cursor is (call it  $B$ ). With the knowledge of these two points we can calculate rotation  $R_i(A, B)$  that takes  $A$  into  $B$ . Fig. C.1 illustrates our current situation. To build the rotation matrix we need to know another

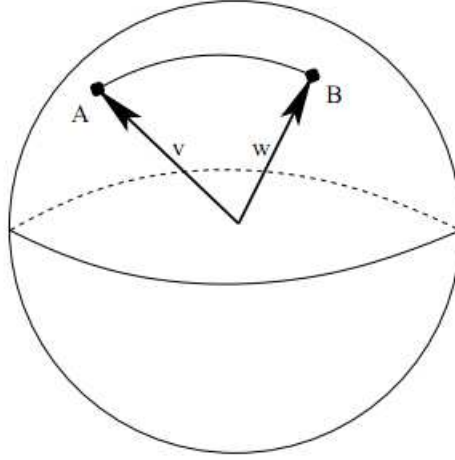


Figure C.1: Two points  $A$  and  $B$  on the trackball

two parameters - *rotation axis* and *rotation angle*. To get the rotation axis, we get the cross product of the two vectors from the centre of the ball to  $A$  and  $B$  -  $\vec{v} \times \vec{w}$ . The rotation angle is an angle between the vectors  $\vec{v}$  and  $\vec{w}$  and is defined as

$$\angle(\vec{v}, \vec{w}) = \arccos\left(\frac{\vec{v}}{|\vec{v}|} \cdot \frac{\vec{w}}{|\vec{w}|}\right) = \alpha\tag{C.5}$$

If we denote  $\vec{u}(x, y, z) = \vec{v} \times \vec{w}$  and normalize it, the rotation matrix has a form

$$\begin{pmatrix} 1 + (1 - \cos \alpha)(x^2 - 1) & -z \sin \alpha + (1 - \cos \alpha)xy & y \sin \alpha + (1 - \cos \alpha)xz \\ z \sin \alpha + (1 - \cos \alpha)xy & 1 + (1 - \cos \alpha)(y^2 - 1) & -x \sin \alpha + (1 - \cos \alpha)yz \\ -y \sin \alpha + (1 - \cos \alpha)xz & x \sin \alpha + (1 - \cos \alpha)yz & 1 + (1 - \cos \alpha)(z^2 - 1) \end{pmatrix}$$

We know how to compute one rotation from  $A$  to  $B$ . This is only sufficient if the rotation is started from the default position. If we already made other rotations prior to this one, we need to track all preceding rotation matrices. If  $R_i(A, B)$  is an  $i$ -th rotation matrix, then the total rotation matrix after  $n$  rotations is given as  $R = R_n \cdot R_{n-1} \dots R_2 \cdot R_1$ .

## C.5 Marr filter

Marr filter is a one of the edge enhancing filters based on the convolution of original image and convolution kernel. Marr filter is based on another edge enhancing operator, the Laplacian. The Laplacian is a 2D isotropic measure of the  $2^{nd}$  spatial derivative of an image. The Laplacian  $L(x, y)$  of an image with pixel intensity values  $I(x, y)$  is given by

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (\text{C.6})$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. Two commonly used small kernels are shown in Tab. C.1. Using one of these kernels, the Laplacian can be

0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

Table C.1: Two kernels for approximating the Laplacian

calculated using standard convolution methods. Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian smoothed before applying the Laplacian filter. This pre-processing step reduces the

high frequency noise components prior to the differentiation step. Gaussian function in two dimensions is defined as

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (\text{C.7})$$

In fact, since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first of all, and then convolve this hybrid filter with the image to achieve the required result. This new operator is called *Marr operator* or *LoG (Laplacian of Gaussian)*. 2D LoG function is given by

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[ 1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (\text{C.8})$$

A part of the discrete kernel that approximates this function (for a Gaussian  $\sigma = 1.4$ ) is shown in Tab. C.2.

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Table C.2: Discrete approximation of LoG operator for  $\sigma = 1.4$

# Appendix D

## Gallery



Figure D.1: High quality skull image with a noticable sinus



(a) View from the top of the body



(b) View along the body

Figure D.2: Two different views at a steel rod in a hip joint



(a) Side of the skull



(b) Back of the skull

Figure D.3: Two different screens of the skull