# Optimization of an advanced algorithm for bone imaging

Adam Mrozek

2022

Master's thesis in Biomedical Engineering

Supervisors:
Professor Hanna Isaksson,
Doctor Lorenzo Grassi,
Doctor Sami Väänänen

LUND
UNIVERSITY

Faculty of Engineering LTH
Department of Biomedical Engineering

Supervisor: Prof. Hanna Isaksson, Dr Lorenzo Grassi, Dr Sami Väänänen

MSc Thesis

Faculty of Engineering LTH
Department of Biomedical Engineering
Lund University
Box 118
SE-221 00 LUND
Sweden


Lund 2022

# Abstract

Fracture risk in osteoporosis is increased and osteoporosis is the most common metabolic bone disease and treatment is expensive. Osteoporosis is diagnosed by measuring Bone Mineral Density (BMD), which does not consider other factors such as bone geometry or architecture. 3D-based models can predict risk fracture better than BMD, but they require 3D computed tomography. Solution of the problem is based on 2D to 3D reconstruction to obtain the 3D geometry. The solution needs considerable amount of time to compute the result and the aim of this thesis is to make the tool perform faster. The workflow for the solution: DXA image of real bone is loaded into memory as the real DXA image. The DXA images come from another study and are already available. The solution contains a set of 3D models of femur and one of the models is selected. A new virtual surface is created and one of the 3D models is projected onto this surface, which results in virtual DXA image. The projection can be done in many different ways, this approach uses a modified version of method called ray-casting. The virtual and real DXA images are compared and the 3D model is changed. A new virtual DXA image is created by projecting the 3D model. The two images are compared and the process repeats until the absolute difference between the real and virtual DXA images is smaller by threshold value. The method was based on analysis and refactoring the existing solution developed by [19] in order to gain increase in speed. The thesis is based on proposal that the program which performs the operation can run faster after applying refactoring of the Matlab code and other techniques. The method is based on rewriting and refactoring the module responsible for projection of the 3D model and creation of the virtual DXA image. The projection of 3D model to virtual DXA image was chosen as the place for refactoring of the code because Matlab tools indicate that this module takes the most of the time during the reconstruction process. Parts of the code were rewritten to make use of different Matlab functions. The largest performance improvement was obtained by providing additional information to Matlab functions, which the code would otherwise need to precalculate on each execution. The performance improvement was achieved by changing the way the solution works with the computer memory. The most favorable result shows that the refactored version can run projections 31% faster than the original version,

but the result is not observable for the reconstruction of the bone. The results were confirmed by tools provided with Matlab. The list of areas which need further research include: other methods of creating virtual DXA image, such as attenuation field, attenuation box, removal of data points to gain speed up and reducing the accuracy, implementation of the solution on GPU. Additional feature which needs implementation is automated landmark placement.

# Abbreviations

- BMD - Bone Mineral Density
- aBMD - Areal Bone Mineral Density
- vBMD - Volumetric Bone Mineral Density
- DXA - Dual Energy X-ray absorptiometry
- GA - Genetic Algorithm
- GPU - Graphics Processing Unit
- PC - Principal Component
- PCA - Principal Component Analysis
- QCT - Quantitative Computed Tomography
- RNG - Random Number Generator
- SAM - Statistical Appearance Model
- SM - Statistical Model
- SSAM - Statistical Shape and Appearance Model
- SSM - Statistical Shape Model

# Acknowledgments

Foremost I would like to give thanks to my supervisors Prof. Hanna Isaksson, Dr Lorenzo Grassi and Dr Sami Väänänen for assisting me in this project. Finally, Dr Lorenzo Grassi: your help can make great difference in how students perceive the world of academics. I would like to thank Prof. Hanna Isaksson for enrolling and supervising me on the project and her valuable comments about the work. Dr Sami Väänänen provided me with very much assistance and endless consultations on the MATLAB source code we have been working with and wrote documents helpful to explain the ideas behind the project. Thanks to Tomas Notermans, Isabella Silva Barreto, Joeri Kok, Elin Törnquist and the rest of the Biomechanics Group at Lund University for always interesting explanations and discussions on bone modeling and mechanics and including me in the team. Special thanks for my family Artur and Barbara Mrozek and friends.

# Contents

# 1

# Introduction

In osteoporosis the fracture risk of bones is increased [23] [3] [8] and over 200 million people worldwide have osteoporosis [1] [9]. Osteoporosis is the most common metabolic bone disease, and osteoporotic fracture considerably decreases life quality. The cost of osteoporotic fractures in Europe will rise to 76.8 billion by the end of 2050 [8] and aging population generates more patients. Therefore, osteoporosis management is important.

Osteoporosis is diagnosed by measuring bone mineral density (BMD) using Dual Energy X-ray Absorptiometry (DXA). The BMD technique is two dimensional and is characterized by the following problem: the prediction of fracture risk using this method does not consider factors that could affect the measurement and consequently the diagnosis, such as bone geometry or architecture [5] [17].

3D-based personalized finite element models of the hip can predict fracture risk better than BMD-based methods [5] [17]. However, such models typically need a 3D computed tomography image to be generated, whilst DXA images, which are 2D images, are commonly available in a real clinical scenario.

The idea is to use 2D-to-3D reconstruction to obtain the 3D geometry and material properties of a specific bone from its 2D radiograph. The reconstruction software was developed in collaboration between the Bone and Cartilage Research Group (Applied Physics, University of Eastern Finland) and the Biomechanics group (Department of Biomedical Engineering, Lund University). The software can perform successful reconstruction from 2D to 3D based on similar 3D models. The reconstruction algorithm takes the DXA image and converts it to a three-dimensional model. The reconstruction is based on a set of three-dimensional models obtained from a scanner. These models are deformed and projected onto a virtual surface and the virtual projection is compared with the DXA image. The process repeats until the virtual projection is similar enough to the real DXA image.

The current implementation [20] has proved to be able to reconstruct 3D femoral anatomies. The difference in BMD between the reconstructed shape of cadaver femur and original QCT (Quantitative Computer Tomography) of a human body from the CT in hospitals (mean $\pm$ standard deviation) varies from 120 $\pm$ 8 to 194 $\pm$ 122 $\frac{mg}{cm^3}$ (from 6% to 66%) depending on the study [19]. The accuracy is satisfy-

ing, however at the moment the calculation speed is the limiting factor. Currently the reconstruction takes several hours, whilst the clinical assessment of fracture risk takes about 15 minutes. Thus, the task approached in this thesis is to identify ways to speed up the reconstruction of the femur.

# 2

# Aim

The aim of the project is to improve speed of the tool used for 2D to 3D reconstruction. The aim shall be accomplished by trying the following solutions:

- Identifying and improving bottlenecks of MATLAB code
- Rewriting and recompiling the code to C++

# 3

# Theory

## 3.1 Introduction

This work is focused on the tighbone (latin *femur*). The 3D image of the bone has been processed by many computer algorithms. The result of the processing is the data about the femoral bone which can be used in the research if appropriate ethical permission is obtained. The processing of the data consisted of the following steps:

- Performing Principal Component Analysis
- Loading the Statistical Shape and Appearance Model
- Obtaining the Digitally Reconstructed Radiographs

The execution of these steps create a 3D model of the femur. The model contains the information about the appearance, shape, density of the bone and can be used in research for many purposes, including the estimation of bone mineral density and osteoporosis treatment.

## 3.2 Principal Component Analysis

The dimensionality (size) of the dataset in the thesis can be reduced. The aim of Principal Component Analysis is to change basis of the data set. The dimensionality reduction can be achieved via PCA by using the first principal components. Suppose that plot of 50 observation on two variables $x_1$ and $x_2$ is given, see [Figure 3.1] and that the dimensionality of this dataset needs to be reduced, for example from 2D to 1D. Replacing a pair of variables $x_1$ and $x_2$ alone will not be successful, however it shall be noted that the points are scattered arbitrarily closely to a straight line and the same set of data with respect to $z_1$ and $z_2$ and variation in the data can be accounted using single dimension. It is possible to recover the two variables using one dimension only. The recovery or decompression of the data works thanks to the fact that instead of $x$ and $y$ components only one variation is saved. Given all

variation the data can be recovered with 100% accuracy. The smaller number of principal components the smaller is the accuracy.

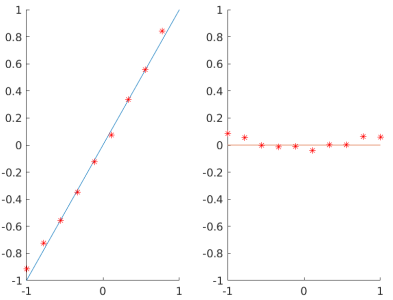


**Figure 3.1** Left: A plot of 50 observations of two variables: $x_1$ and $x_2$. For example, $x_1$, $x_2$ respectively might represent volume and density of the bones. Right: Plot of the 50 observations with respect to their Principal Components $z_1$ and $z_2$. Figures adapted from [10]. Note that the $y = x$ to the left became the x-axis to the right by rotating the figure.

Applying Principal Component Analysis to any set of data results in **coefficient matrix**. The coefficient matrix $A$ is used to exchange data between the components of the program used in this thesis. To optimize the speed of the program one might want to regulate the amount of data the computer needs to calculate. One of the ways of doing this is varying the number of Principal Components. The point from the mathematical point of view is to standardize $x_1, ..., x_n$ by dividing each by its standard deviation and finding linear functions of these standardized variables which maximize variance [10].

- The first eigenvector consists of coefficients $a_{11}, a_{12}, ..., a_{1p}$ and is the first Principal Component.
- The first eigenvalue is the variance of the first Principal Component etc. In practice the Principal Components are found from correlations (or covariances) between $x_1, ..., x_n$.
- The second, third, ... eigenvectors are the second, third Principal Components and so is for the eigenvalues

The first eigenvalue contributes with the most variation and has essential role for recovering the shape of the bone, whilst the last eigenvalues contribute with least variation. As a consequence, removal of the least significant eigenvalues would result in slightly decreased accuracy and could reduce the amount of data **possibly**

**increasing the speed of the computations** (see [Figure 3.2]). This removal is also the main idea behind the dimensionality reduction that can be achieved via PCA. A fraction of the components can be used to achieve accuracy which is very high, albeit never 100% if any of the modes were rejected.

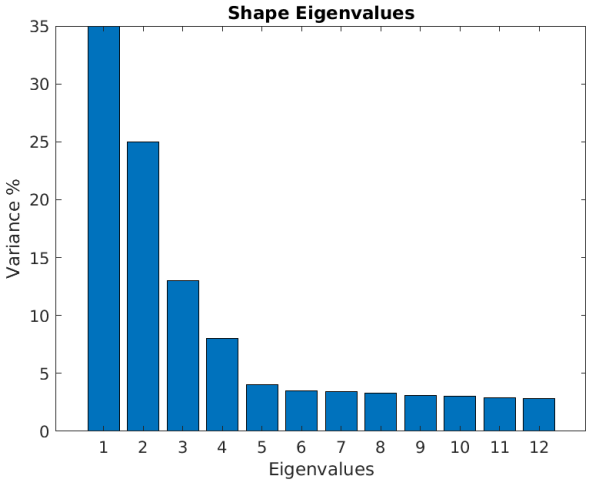


**Figure 3.2** The first eigenvalue, i.e. mode contributes with the most variation and consequently the amount of information about the bone. Last eigenvalues are of least significance and can be omitted in certain application.

## 3.3 Statistical Shape and Appearance Model

Statistical Shape and Appearance models provide information such that the reconstruction algorithms have data to base their decisions upon, otherwise it is not possible to estimate how an average 3D shape would look like. SSAM relies on the silhouettes (shapes and outlines) and the appearance of the complete anatomical structure.

Statistical Shape Models and Statistical Appearance models are two statistical models:

- Statistical Shape Model describes the average shape with the main modes of variation of shape distribution within a population
- Statistical Appearance Model describes the average density distribution with the main modes of variation within a population.

These two models combined in one model is the Statistical Shape and Appearance Model (SSAM).

SSAM is the output of the PCA process and creates a powerful tool for this project, because the modes allow the change of the statistical appearance of the bone quickly. Changing value of the first mode contributes to the largest changes of the shape of the bone and the last modes contribute with least change. [Figure 3.3] show the difference between two models of the same bone with modes altered. [Figure 3.4] is an illustration of the femoral bone as seen in anatomical atlas compared to the 3D representation seen by the computer.

Some reasons why this approach is used and the advantages of PCA over storing the components as $(x, y, z)$ coordinates are:

- Ease to modify the shape and appearance of the bone by changing coefficient matrix $x$ in $Ax$, where $A$ is the covariance matrix [Figure 3.2].
- The modes of variation are already sorted by overall variation in bone anatomy they contribute to, the most variation at the beginning, the least variation at the end, at no additional computational cost.
- The model of the bone can be obtained by adding contributions of a limited number of the principal modes of variation to the average shape and density distribution.

The disadvantage is:

- The data do not always come in the form of coefficient matrix $A$. In cases where the bone data is not contained in the SSAM, conversion to the coefficient matrix must be performed.

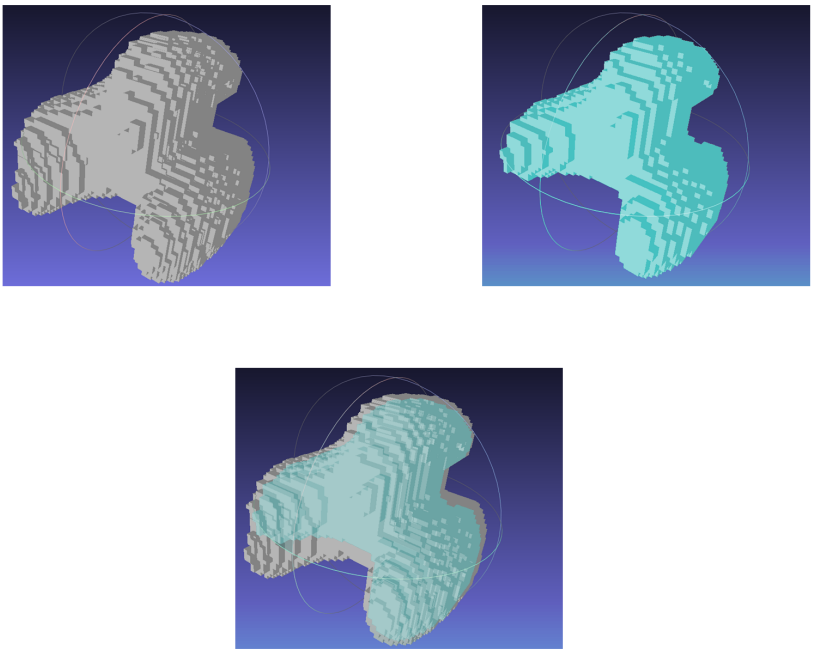

**Figure 3.3** Femur with different coefficient matrix $x$. The coefficient matrix $A$ is the same for all shapes. The greater trochanter (*trochanter major*) and femoral head (*caput femoris*) and the bone shaft differ in size to show how the change of modes changes bone appearance. The image to the left and the image to the right show the same bone with different principal modes. The figures have been colored and merged to clarify the anatomical differences. See [Figure 3.4] to see the anatomical structure represented by this model.

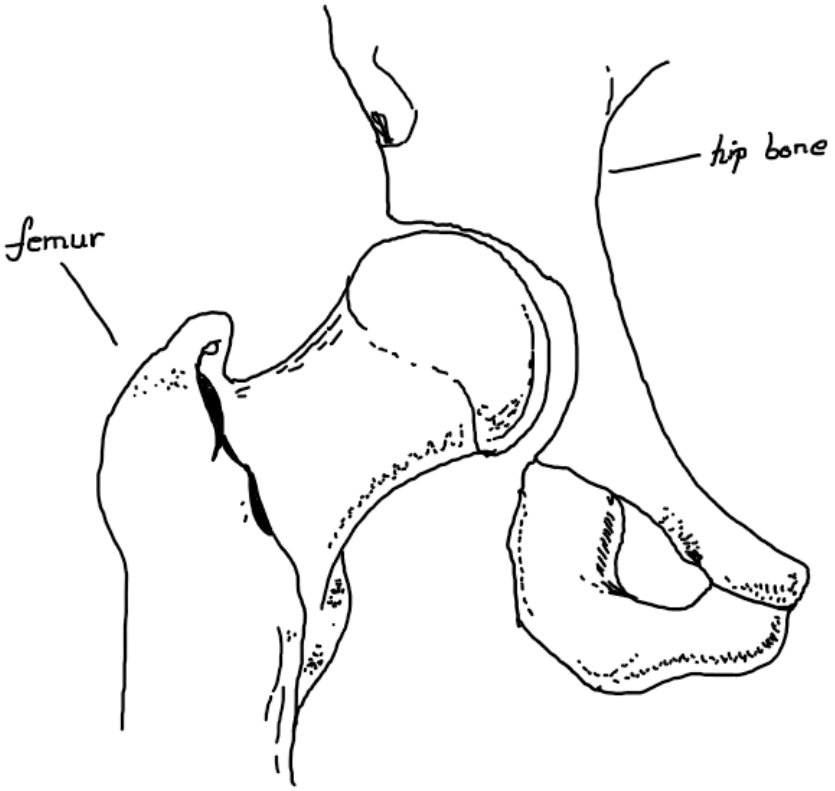


**Figure 3.4** Femur - the main subject of the research. The bone is segmented and analyzed by the computer. [Figure 3.3] presents the bone seen by the computer program. Image based on [16]

## 3.4 Digitally Reconstructed Radiographs

A digitally reconstructed radiograph is a perspective projection of the three-dimensional image onto a 2D image plane [6] [Figure 3.5]. Digitally Reconstructed Radiographs (DRR) are simulated radiograph images calculated from projection of 3D images, such as computer tomography (CT) or magnetic resonance (MR) [6]. It is an image in two dimensions, looking like an X-ray image. The difference between an X-ray image and a digitally reconstructed radiograph is that the later does not need a three-dimensional CT or MR scans. Instead it can use any 3D mesh with its density function to create a virtual radiograph. The classical approach used in the thesis is modified ray-casting. The other common algorithms to generate DRR are described later in this section [Figure 3.5]. Generation of Digitally Reconstructed Radiographs is computationally expensive and is typically the limiting factor in the process of 2D to 3D reconstruction [13].

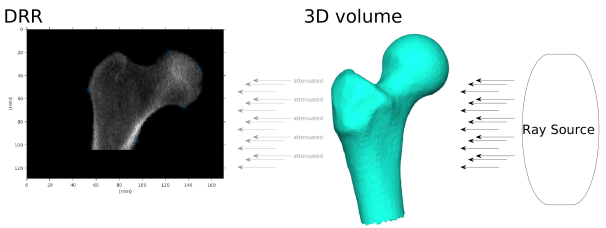


**Figure 3.5** Ray-casting is the traditional method for the generation of Digitally Reconstructed Radiographs

There are many methods to create the Digitally Reconstructed Radiograph. Some of these methods are listed in [Table 3.1]. The methods are summarized in the table below, and brief albeit more detailed description follows after the table.

| Method | Short Description |
|---|---|
| Raycasting [19] | Shoots rays from virtual ray source onto virtual surface through the three dimensional model. The analysis of this method was performed in the thesis. |
| Attenuation Field [13] | Performs preprocessing of most of the DRR computations, but the generation of the DRR is faster |
| Attenuation Box [6] | Based on Attenuation Fields [13], the process is accelerated using geometric observations |
| Splatting method | Each voxel is projected individually but the distance between the voxels increase deteriorating the quality of the DRR. |

**Table 3.1** Summary of some of the methods for generation of the Digitally Reconstructed Radiographs

## Ray-casting

There are many different ways to generate a DRR. The traditional method for the generation of DRRs is called ray-casting. This method was implemented by [19] and is used in this research. The solution in [19] creates a virtual radiograph from the three-dimensional model. For this approach, the two components are essential:

- Virtual Ray source
- 3D Model

The ray is released from the virtual ray source and travels through the 3D model. The ray attenuates depending on the density of the 3D model. As the model contains information about the density, the attenuation is different for each point in space [Figure 3.6]. In this method, the value of each pixel in the DRR is computed by sampling intensity values along the ray connecting the ray source to the pixel [Figure 3.5]. The ray-casting approach used in the project by [19] generates the exact projection. The main disadvantage is that the process takes time.

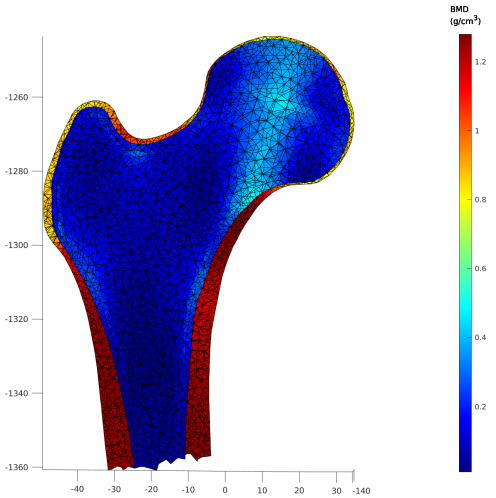


**Figure 3.6** Each point in the three-dimensional model contains information about its BMD. Picture generated by software in [19]

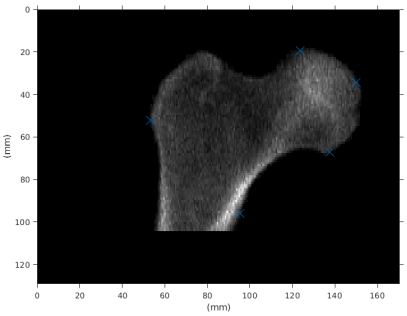


**Figure 3.7** Illustration of a Digitally Reconstructed Radiograph (DRR). The real radiograph is characterized by the same appearance.

## Attenuation Fields

According to [13], Attenuation Field originates from technique called light fields and allows most of the DRR computation to be performed in a preprocessing step and not during the generation of the DRR, which increases the speed of computations.

The preprocessing step needs only to be performed once, and after it the generation of the DRR will be faster, as computations were performed during the preprocessing step. The experiment performed by [13] shows that the traditional approach using Ray-casting takes 3000 s whilst the Attenuation-field accelerated approach takes only 100s. No information is given about the preprocessing time.

### Attenuation Box

This section contains a brief description of the method called Attenuation Box. According to [6], who based their research on [13], the Attenuation Box is an improved version of the Attenuation Field method.

The Attenuation Box is a three dimensional box surrounding the three-dimensional model. The method is based on the observation that each ray which contributes to the projection can be uniquely described by the set of two intersection points with the cube (the attenuation box). The resolution of the grid on the attenuation box allows to adjust the resolution and quality of the DRR. The idea of the attenuation box is shown on [Figure 3.8].

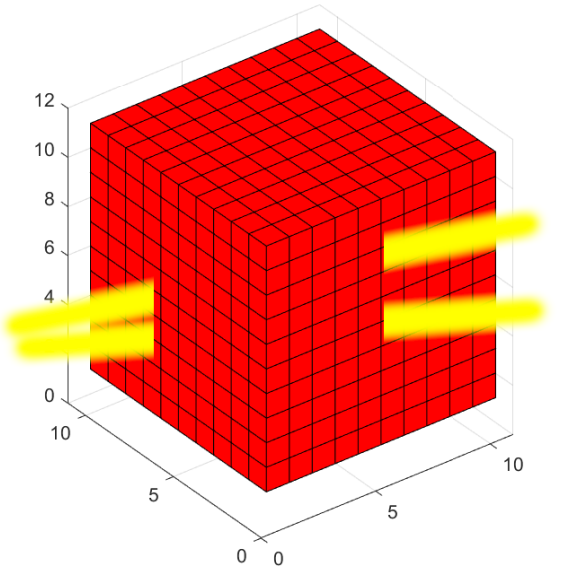


**Figure 3.8** Illustration of an Attenuation Box. Computer memorizes the points where rays cut the box and uses this data to prevent unnecessary computations. The model is hidden inside the box.

## Splatting

The ray-casting and its derivatives (Attenuation Fields and Attenuation Box) are image space volume rendering methods as opposed to the splatting method. The splatting method is based on advanced algorithmic techniques as object space DRR rendering or Maximum Intensity Projection (MIP) [21]. In MIP each voxel (pixel) is considered as self-emitting light source and the final pixel intensity is computed as the maximum voxel value among all voxels projected on the reconstructed DXA image [2].

## 3.5 Genetic Algorithm

A genetic algorithm is any model which uses selection and population to generate new set of solutions. This algorithm will repeat until the set of solutions approximates the correct solution well enough. By well enough is meant that the difference between the recreated radiograph and the DXA image is smaller than some value $\varepsilon$. A genetic algorithm starts off with a population of random individuals of chromosomes. Individuals which produce a solution which minimizes the difference between algorithms output and reference output have more opportunity to reproduce.

To understand the GA, consider a parameter optimization problem; the difference between the reconstructed and real radiographs must be minimized and different parameters give different reconstructions of radiographs. In GA a population of individuals will be created and each individual performs a SSAM based reconstruction of a 2D radiograph. The radiograph is projected onto a surface and a virtual radiograph is obtained. This virtual radiograph will be compared to the original radiograph. The reconstructions and their projections which are most similar to the real radiograph create the next generation and iteration of the GA. An example of how the GA can reciprocate into next generations is shown in [Figure 3.9].

To use the genetic algorithm two problems must be solved: *problem encoding* and *evaluation function* [22]. In this project the problem is encoded using coefficient matrix with principal components. Problem Encoding is how the problem is seen by the GA and evaluation function is how the GA will solve the problem. The evaluation function used in this project is a modified version of ray-casting.

There are four important parameters for the GA:

- `popSz` - population size for the Genetic Algorithm
- `nofGen` - maximum number of allowed generations
- `funTol` - the minimum improvement between the generations. If an iteration improves by less than `funTol`, the GA will terminate and assume that the optimal solution was found
- `stallGen` - the maximal amount of generations without improvement before GA is considered converged

The genetic algorithm stops when the solution cannot be improved by more than a value defined in the implementation. In this project the sum of the absolute differences between reference radiograph and the virtual projection of the reconstruction (virtual radiograph) must be smaller than `funTol`.

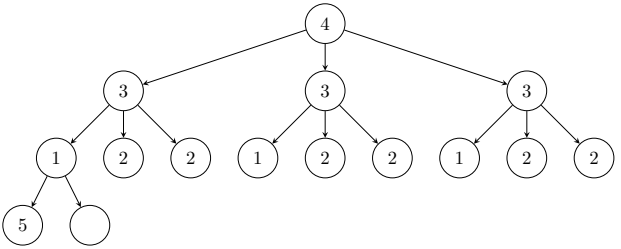


**Figure 3.9** Example progression of genetic algorithm.

## 3.6 Reconstruction is a feedback loop

The reconstruction algorithm is based on a feedback loop. Reconstruction of a 3D subject from multiple or single radiograph is possible. If one radiograph is given, so is in this case, SSAM is what gives the algorithm an idea of the anatomy of the body part. This way both shape and appearance are used to describe the model and therefore Statistical Shape and Appearance model combines Statistical Shape model and Statistical Appearance model [14]. For each iteration of the loop a better approximation of the solution is calculated.

# 4

# Material and Methods

The following section describes materials and methods used in the research.

## 4.1 Materials

The materials consist of Matlab solution developed by [19] at the University of Eastern Finland, materials from the Biomechanics Group at Lund University including the source files and models of bones. The unpacked project according to [7] consists of over 100 thousand lines of code. The project uses library `iso2mesh` [19], which allows to perform many advanced operations on 3D objects. The project needs the three components to run:

- the program
- DXA images for reconstruction
- SSAM

The architecture overview for the project is presented on [Figure 4.1]. The DXA images were obtained from the Biomechanics group at Lund University.

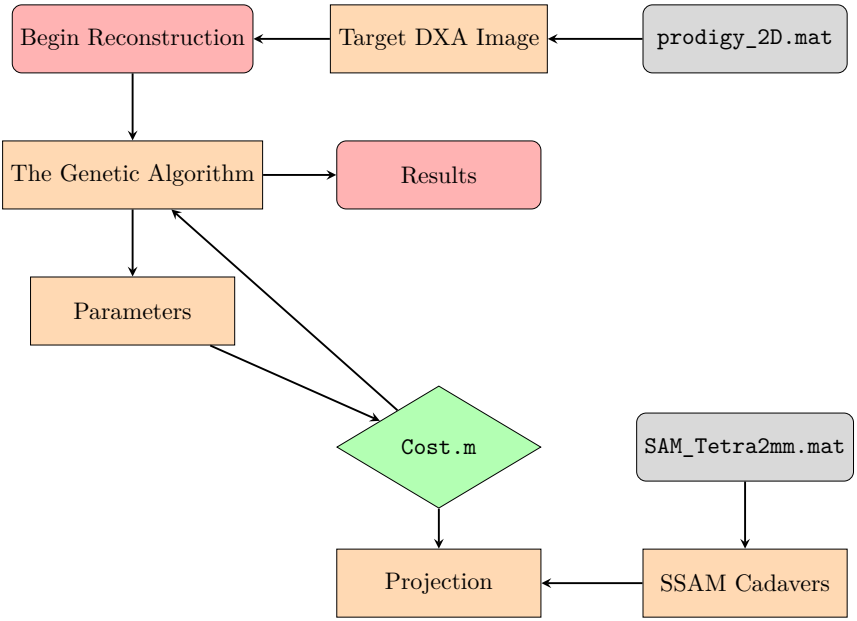


**Figure 4.1** Overview of the project architecture

## Overview of the source tree

The program files contain a set of radiographs together with the SSAM. The radiographs are found in file `prodigy_2DimageObj.mat` in the project directory. The Matlab file contains 37 DXA images with landmarks. The images were obtained using the Prodigy scanner and the research began with the data already processed by the scanner. The file contains cell arrays, a Matlab data structure and each cell contains a DXA object. A DXA object inherits from Matlab class `Image2D`, which inherits from `image`. The DXA object contains the real radiograph with landmarks. The landmarks tell the characteristic positions of the bone. The first image (CAD001) from the Prodigy data set is shown in [Figure 4.2] and there are 38 such images.

Matlab Coder and Matlab Profiler were used. Matlab profiler allows to measure the time it takes to run the code and identify where the software spends the most time. The Matlab Coder generates C and C++ code from Matlab code for many hardware platforms [7].

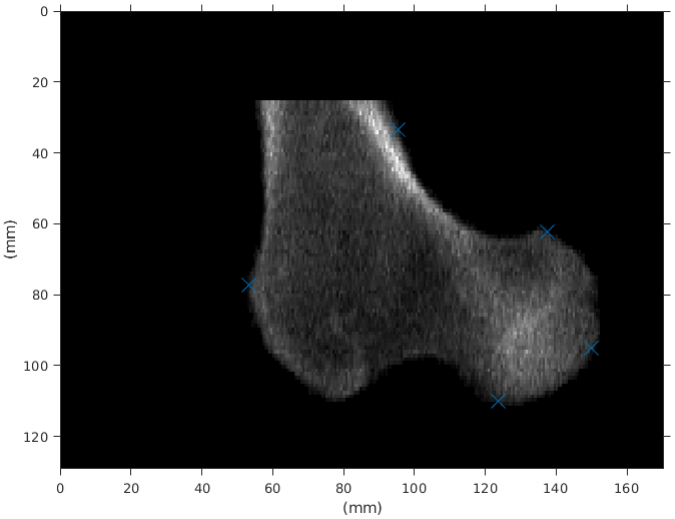


**Figure 4.2** The first radiograph from the set of radiographs with landmarks marked in blue

The materials include a broad collection of Matlab scripts and functions which perform SSAM-based 2D-to-3D reconstruction. The steps for preparing the full reconstruction are:

- Setting parameters for the GA
- Loading DXA objects from file `prodigy_2DimageObj.mat`
- Creating Cost object
- Loading SAM object from file `SAM_Tetra2mm`
- Preparing input for the GA
- Running the reconstruction algorithm

The GA then sends the data to the projector, which creates the DRR. The DRR is sent back to the GA and is compared against the real radiograph of the patient. The creation of the DRR follows the procedure:

- The function `calculatecost` (Cost Function) is called by GA.

- Inside of the `calculatecost` SAM instance is obtained by calling `getSAMinstance`.
- The SAM instance is converted to a mesh and then the mesh consisting of tetrahedrons is converted to a 3D image, where each pixel represents the BMD in that point.
- The image is sent to the function `floatingmesh2image` and then `mesh2image` where the DRR is built by a method similar to ray-casting.
- The radiograph is sent back to the Cost function, which returns the difference between the QCT real and DRR virtual radiograph to the GA. The GA then modifies the parameters and starts with the Cost function again.

A file which is a part of the 2D to 3D reconstruction in the implementation by [19] called `mesh2image.m` is an important Matlab script and was the subject of the optimization. The script receives parameters which describe the three dimensional model and produces the virtual radiograph by DRR.

## 4.2 Profiling and refactorization

Using the genetic algorithm to create a population of individuals the script tries to obtain a correct 3D model. Each individual creates children, where each children receives modified parameters. The children with the best results create more sub-processes, and the least effective generations disappear. In the example [Figure 3.9] an example tree is shown.

For each of the nodes in the tree shown above, an iteration of the projection algorithm is executed. The projection is the slowest method of the computer program as shown by Matlab Profiler. The projection algorithm converts a 3D mesh to the digitally reconstructed radiograph using raycasting. The original algorithm is described in pseudocode in [Algorithm 1].

The projection algorithm creates the digitally reconstructed radiograph from a set of cadavers. The behavior of the script is configurable by five parameters:

- `popSz` - population size
- `nofGen` - number of generations for the genetic algorithm
- `funTol` - stopping criterion for convergence, for example 10e-20.
- `stallGen` - maximal number of stall generations
- `vararg` - if available, the method can be provided with preprocessed data about the mesh by this argument

**Algorithm 1** Mesh2image in original

```
1:  procedure [(M)esh2image]node, elem, intensity, vararg▷ Converts a mesh to a
    3D matrix
2:      intensitytype ← node, elem, node + elem          ▷ Need preprocessing?
3:      switch intensitytype do                  ▷ Perform necessary preprocessing
4:          case node + elem
5:              Continue;
6:          case node
7:              ComputeIntensity;
8:          case elem
9:              ComputeNodes;
10:     nodemin ← coordinate of minimal node          ▷ Find boundaries of the mesh
11:     nodemax ← coordinate of maximal node
12:     nodetranslation ← padding necessary for image operations
13:     node ← node + nodetranslation                                ▷ Move the mesh
14:     switch intensitytype do          ▷ Precalculate intensity values if necessary
15:         case node + elem
16:             nodeintensity, elemintensity ← node, intensity
17:         case node
18:             nodeintensity ← nodeintensity(elem)
19:             elemintensity ← estimateIntensity(node)
20:         case elem
21:             Estimate intensity using advanced methods
                                                                  ▷ MeshToVoxels
22:
23:     elcent ← all vertex coordinates and element centers
24:     elcent ← round(elcent)
25:     nodeidx, elcentidx ← vertexandelementcenterin3Dimage
26:     im3D ← nodeintensity, elemintensity
27:     im3D ← Morphologically close image (im3D)
28:     im3D ← Compute Euclidean Transform of the image (im3D)
29:     Do projection if configured to do so
30:     coord ← Non zero elements of im3D3
31:     if Only coordinates requested then return coord
32:     end if
                                  ▷ Find all vertices coordinates and element centers
33:                       ▷ Find vertices and element center indexes in 3D image
34:                         ▷ Return image values and coordinates of each voxel
35:
36:     coord ← findn(im3D3)                                         ▷ Make image
37:
38: end procedure
```

A stall generation means that no improvement of the result was observed. If the script does not see any improvement it will stop and this variable specifies how many generations it needs to wait before it can be certain that no further improvement of the results is likely to appear.

### Refactorization using Matlab Profiler

The tool used to optimize and profile the code is included in Matlab. The Profiler was used to measure the time it takes to run the code. Later the Profiler was used to indicate which parts of the code take the most time to execute. The method (as in Matlab documentation) consists of the following steps [7]:

- Running the profiler on the code
- Reviewing the profile summary results
- Investigating functions and individual lines of code.
- Saving the profiling results
- Implementing potential performance improvements in the code
- Saving all the files and re-run the profiler to compare the results to original results
- Repeating all the steps and continue improving the code until most of the execution time is spent on built-in Matlab functions

According to the documentation the following steps were performed:

- The reconstruction was started for different configurations of the input parameters of the genetic algorithm. The reconstruction was measured for various number of generations, some of them are reported in the results section, but the number of measurements was more than 270.
- Using Matlab Profiler the functions that consume the most time were identified. Profiler generated a summary of functions and the results were reviewed. Profiler is enabled by appropriate function calls in Matlab.
- Candidate functions were evaluated for possible performance improvements using static code analysis. The most focus was put on functions which use a significant amount of time or are called the most frequently. The runtime was measured multiple times.
- From all the functions, the `findn` was identified to take the most time and an attempt was made to rewrite the code around it. The code was rewritten with focus on speedups. The purpose of the `findn` function is to identify the nonzero entries of the data structure passed as input. When the program stars the

analysis of the proximal femur, the function `mesh2image` is executed during the analysis.

- Profiler ran again to compare the results to the original results. The process repeats until the function runs fast enough.

Three solutions were implemented, these are:

- Reimplementation of `findn` and rewriting functions using `ind2sub`
- Generating C++ code for the project

### Refactorization by reimplementation of `findn`

According to Matlab documentation [7], the `findn` function finds the indices of nonzero elements of the input argument $X$. The input argument $X$ is an array.

The first implementation of `findn` was rewritten to the following program shown in pseudocode. The reimplementation was necessary to prepare the solution for compilation into C++. This computationally ineffective implementation only works with 2D and 3D data as these are the only dimensions required for the project.

```
function oout=my_findn(arr); %for codegen

in=find(arr);
sz=size(arr);

if nDim == 2
    oout = [];
    for i = 1 : sz(1)
        for j = 1:3
            if (arr(j,i) ~= 0)
                oout=[oout;[j i]];
            end
        end
    end
else if nDim == 3
        oout = [];
        for i = 1 : sz(1)
            for j = 1:sz(2)
                for k = 1:sz(3)
                    if (arr(i,j,k) ~= 0)
                        oout=[oout;[i j k]];
                    end
                end
            end
```

```
        end
    end
end
```

### Rewriting `findn` using `ind2sub`

Matlab Code Generator was launched to generate the C++ version of the projection module. Then the code was rewritten again to work as Matlab native code which cannot be easily translated to C++, but uses Matlab-specific functions, like `ind2sub`.

```
function coord=my_findn2(arr)
coord=ind2sub(size(arr),find(arr));
end
```

The code was rewritten again to:

```
coord=zeros(nnz(im3D3),3);
[coord(:,1),coord(:,2),coord(:,3)]=ind2sub(size(im3D3),find(im3D3));
```

The first row of the final implementation is crucial for the results of this work, because it changes the way Matlab manages computer memory. The extra line allocates memory in advance for the execution of the `ind2sub`, which would otherwise need to perform multiple memory allocations taking the extra time to finish. Each memory allocation is computationally expensive and requires exchanging information with the operating system.

## 4.3 Compilation to C++ using Matlab Coder

Modules in the project were rewritten to C++. The refactoring of the project was made using the built-in Matlab coder. The workflow was performed according to Matlab documentation [7]. According to Matlab documentation, the following describes the Code Generation Workflow:

- Set up Matlab Coder Project
- Prepare Matlab Code for Code Generation
- Test MEX function
- Generate C/C++ code from Matlab Code
- If needed: Optimize generated code and go back to Prepare Matlab Code for Code Generation
- Deploy Generated Code

These steps were performed to create the native executable file from C/C++. The code was prepared for code generation by removing functions and converting data types which are not supported by the Matlab Coder. In particular:

- Data Structure called Cell Array works only in Matlab and is not easily translatable to C/C++.
- Arrays with undefined size

In order to solve the issues Cell Arrays were removed and replaced by arrays or vectors. Arrays with undefined size were converted to constant-sized arrays in C++. Built-in Matlab functions which were not machine translateable were rewritten in Matlab to allow automated translation to C++. The function `findn` was not machine translatable and was rewritten in Matlab. This function finds indices and values of nonzero elements [7]. While the original version would work for all dimensions, the rewritten version would only limit to three or two dimensions as:

- This makes the function machine-translatable to C/C++ as the dimensions of the data set are defined
- The project does not require analysis in more than three dimensions

The generated MEX function was tested against the original Matlab function and an attempt was made to recreate the femur from the C++ generated code. The recreation succeeded meaning that the converted C++ version of the code produces the same results. The optimization techniques used for this approach were:

- Preallocating memory to prevent memory allocations
- Optimizing code to utilize hardware-based CPU instructions

Two measurements were taken during the experiment. The process needs proof that the optimized version of the algorithm performs better than the unoptimized version. The reconstruction takes an amount of time $t$ measured using system clock and commands `tic` and `toc`. The results from the timing test were summarized in graphical form and the variation and mean value was computed. While the tests were running all system services and background applications other than these, which were essential for the operating system were closed down. This is to prevent additional CPU load while the experiment is taking place.

The benchmarks have names and the name of each benchmark tells the size of the population and the maximal amount of generations the algorithm is allowed to execute. The benchmark for the genetic algorithm was executed 250 times to measure its mean and standard deviation for different parameters of population and generation size.

# 5

# Results

## 5.1 Introduction

This section shows that:

- The choice of the function for optimization was a result of profiling and the slowest of the functions was chosen. The slowest function takes up the majority of the program execution time (70% according to the Matlab Profiler).

- The refactorization of Matlab code can improve the speed of the Matlab code and that the performance increase is achieved by using functions which are implemented in a way which is more friendly for the CPU.

- The compilation of the code to C++ might theoretically create an improvement in speed due to refactorization which is necessary to compile the Matlab code to C++.

## 5.2 The choice of functions for refactorization

The choice was based on the results from the Matlab Profiler [Figure 5.1]. The graph is read from top to the bottom. The main function, which performs the entire 2D-to-3D reconstruction is `script_tutorialfor3DreconstructionfromDXA`. Each new function is presented as new rectangle to the right. As time passes new functions are created and the functions create own functions. Whenever a function terminates, its rectangle disappears. The whole reconstruction process consists of:

- Initializing the computer memory in `Cost` block
- Loading the data into memory in `SAM>SAMLoad`
- Creating the DRR using the GA

According to Matlab Profiler the Genetic Algorithm spends 77% of time to make one step of the 2D-to-3D reconstruction. The 92% of the GA is spent on reconstruction of the radiograph, the rest if the time is used to compute new parameters for the next round of the GA and comparing the absolute difference between the virtual and real radiograph. The reconstruction is the most time consuming function written by non Matlab developers and takes over 70% of the time of the reconstruction. The reconstruction process is performed by the GA only. In the figure the GA is presented as the longest gray rectangle labeled `ga`. The most time consuming function of the reconstruction is called `mesh2image`.

### Further analysis of execution time of the `mesh2image`

The most time consuming function of the reconstruction is `mesh2image`. The function consists of the code written by the engineers and authors of the 2D-to-3D reconstruction algorithm [20] and calls to functions developed by Matlab engineers. The analysis of external calls was performed and the summary of the most time-consuming external calls is available in [Table 5.1]. The time spent by the external function calls and the code developed by [20] is presented on the same flame graph [Figure 5.1]. The ratio of the time spent by the external function calls vs the code developed at the institution is 50%/50% as presented on the graph. [Table 5.1] presents the external function calls of the `mesh2image` and briefly states their purpose.

| Function Name | Description | Percent time |
|---|---|---|
| `bwdist` | Transforms binary images | 28% |
| `findn` | Finds indices of nonzero elements | 9% |
| `imclose` | Morphologically closes image | 8% |
| `sub2ind` | Computes linear index from multiple subscripts | 2% |

**Table 5.1** The table of functions called from `mesh2image`, the most time consuming function of the reconstruction


**Figure 5.1** Matlab Profiler shows the time it takes for each function to run. The graph shall be read from top to bottom. Each new child creates rectangle to the right. The most time consuming function of the reconstruction is `mesh2image` and it takes approximately 70% of the GA.

## 5.3 The results of the refactorization

### Speeding up by refactorization of Matlab code

As the function `mesh2image` was identified to take the most time during the reconstruction, it was chosen for refactorization. After the refactorization and optimization the function `mesh2image` speeds up from 0.105s to 0.072 (by 31%) in the Matlab Profiler as can be seen on [Figure 5.2] on reconstruction of single DRR from parameters received from the GA. For the entire and accurate 2D-to-3D reconstruction many executions of the most time consuming function must be performed.

The speedup obtained from refactorization of the Matlab code improved the speed of execution up to 31% for single DRR for parameters received from the GA. The comparison of the original Matlab implementation provided by the faculty and the refactored implementation optimized in the thesis is presented on [Figure 5.2]. There are two measurements taken for each of the tests: Total Time and Self Time. These are the two times measured by Matlab. Total time presents the time difference from when the program starts until the time it is finished its work. Self Time measures the time spent by the program on computation and does not include other tasks performed for example by the operating system and other software the reconstruction runs on.

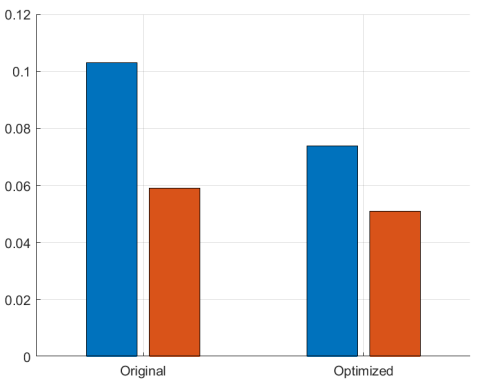


**Figure 5.2** Matlab Profiler presents timings for the original and optimized implementations.

## 5.4 Compilation to C++ using Matlab Coder

The same function was chosen for refactorization and rewritting to C++. The rewritten function was compared to the original and the time spent by the compiled version is over 90% longer than the refactorized Matlab version. There are too two measurements taken for the test: Total Time and Self Time, both measured by Matlab [Figure 5.3].

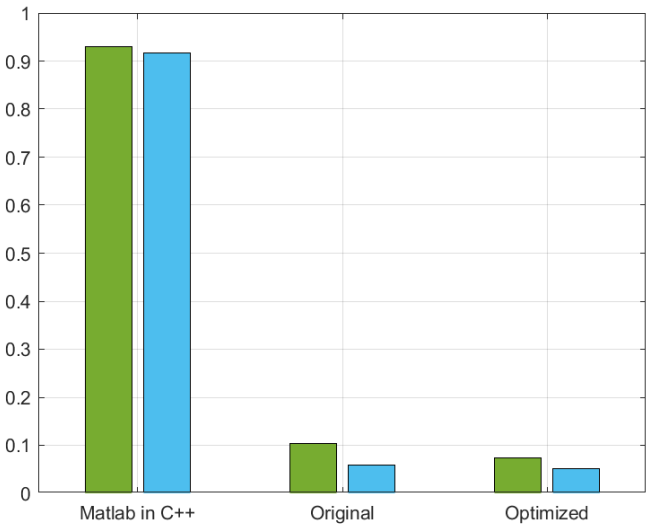


**Figure 5.3** Matlab Profiler presents timings for Matlab and C++ versions of the function. The profiling function provides the two times of execution.

### The analysis of execution time of the whole reconstruction

As described in the theory section, the accurate 2D-to-3D reconstruction is based on multiple reconstructions of the 3D model and is based on a feedback loop with undetermined number of iterations. This section shows that optimizations of the reconstruction algorithm do not bring visible optimizations for the entire 2D-to-3D reconstruction process.

The time to perform the entire 2D-to-3D reconstruction was measured on the optimized version of the algorithm. The result of the measurement shows that the average execution time may vary for the same implementation and the same set of data for algorithms which are dependent on the random factor. The results of the overall runtime for the whole reconstruction, including the iterations of the Genetic Algorithm were summarized. It was not possible to observe improvement in the en-

tire 2D-to-3D reconstruction despite the fact that single reconstruction of the model indeed speeds up.

The five tests were chosen from a test suite of over 270 tests running over days on the test computer. The sample from the results from these test are summarized in [Figures 5.4, 5.5, 5.6, 5.7, 5.8]. A test case consists of five bars representing the same five cadavers, as it is in the original Matlab code. The results are characterized with high standard deviation from 360 to 1000 seconds. The average reconstruction time was 1340 seconds for all test cases for the optimal version of the algorithm.

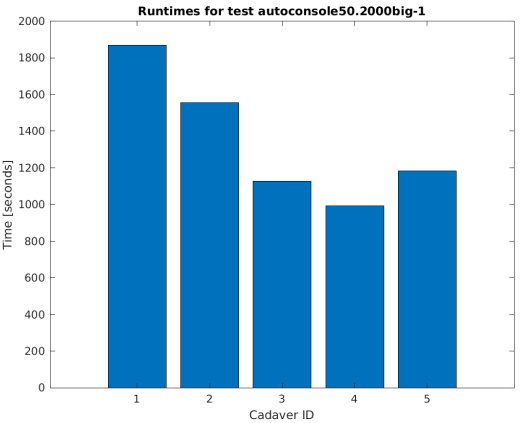


**Figure 5.4** Runtimes for test autoconsole-pop50gen2000-1 with mean $\overline{X}$ = 1.345255e+03 and standard deviation $\sigma$ = 3.600211e+02 . The population size for this test was 50 and the algorithm ran in 2000 generations.

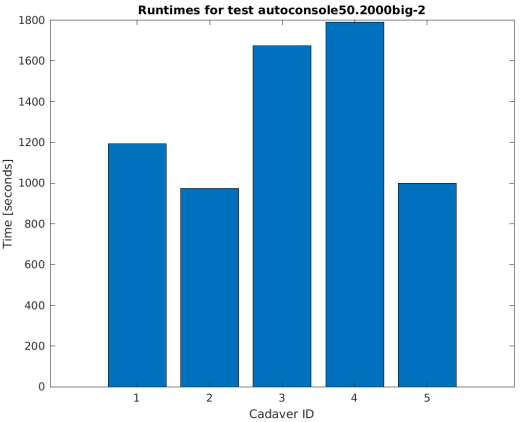


**Figure 5.5** Runtimes for test autoconsole-pop50gen2000-2 with mean $\overline{X}$ = 1.325783e+03 and standard deviation $\sigma$ = 3.824948e+02 . The population size for this test was 50 and the algorithm ran in 2000 generations.

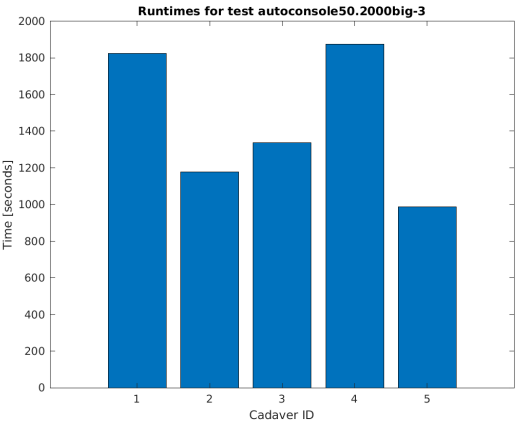


**Figure 5.6** Runtimes for test autoconsole-pop50gen2000-3 with mean $\overline{X}$ = 1.439831e+03 and standard deviation $\sigma$ = 3.942488e+02 . The population size for this test was 50 and the algorithm ran in 2000 generations.

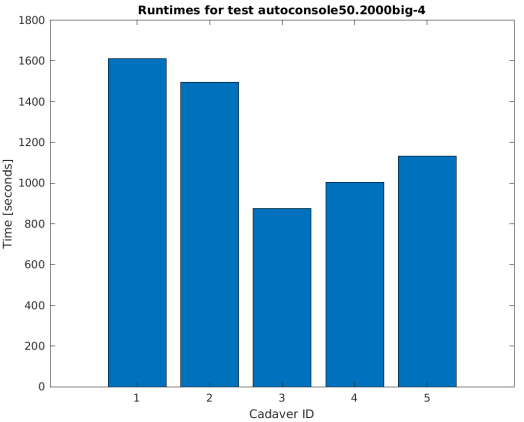


**Figure 5.7** Runtimes for test autoconsole-pop50gen2000-4 with mean $\overline{X}$ = 1.224253e+03 and standard deviation $\sigma$ = 3.161706e+02 . The population size for this test was 50 and the algorithm ran in 2000 generations.

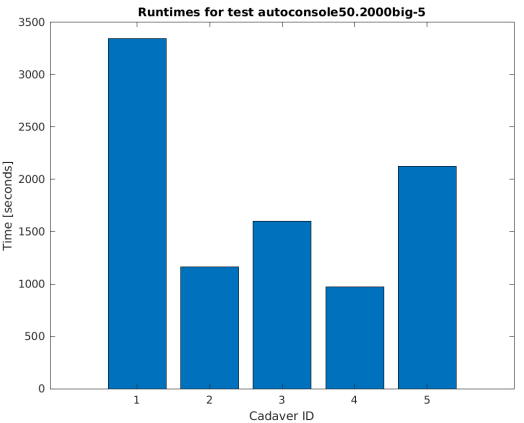


**Figure 5.8** Runtimes for test autoconsole-pop50gen2000-5 with mean $\overline{X}$ = 1.840194e+03 and standard deviation $\sigma$ = 9.494881e+02 . The population size for this test was 50 and the algorithm ran in 2000 generations.

# 6

# Discussion

## 6.1 Which is the right function for optimization

The computer based solution of the reconstruction problem consists of huge amount of code. Therefore some research with Matlab Profiler was conveyed and candidates for optimization were obtained. The question the project was facing was which functions require the most optimizations. To answer this question, the anatomy of the reconstruction program needs to be briefly explained. The analysis does not include the technical details of loading the data into memory as this stage needs always to be executed. When the data is loaded, the genetic algorithm starts the reconstruction algorithm and compares the result of the reconstruction with the real image.

## 6.2 Refactorization of Matlab code

In this thesis a modification of code used for research was done. The running code is faster by 31%. The part of the program which was optimized was the module which is responsible for creating the virtual DXA image from the three-dimensional model. In the beginning of the project more advanced optimizations were planned and therefore it should be possible to further optimize the code. The aim of the thesis was reached. As the code is non-deterministic the results may vary, but the deterministic part of the solution performs better.

Optimization is a broad term and the methodology of the research might appear unusual for engineers who did not become familiar with computer science. Although the method is specified and documented by Matlab, the results of the optimization may vary depending on how many methods are implemented in sub-optimal way. One who attempts the optimization is always aware that the code to be optimized might be the most optimal version of itself and in some rare cases no work can be done in this area. In this project, there are still pieces of code which have potential for optimization and work was done to increase the speed of the program.

There are many methods to optimize the code, many of the common methods when thinking on optimizations are low level methods. Many such methods are described in [4], [15]. Some examples of the mainstream methods described in these publications are:

- Divide the data into sub-problems, solve each subproblem and merge (divide-and-conquer)
- Build a structure in memory to find and modify the data easier (tree algorithms)
- Save the state of computation between iterations (dynamic programming)
- Precompute the values before the actual program runs (preprocessing)

The reason why these methods are universal for many applications is that all computer programs work on microprocessors which are similar in construction [15]. Computer programs work according to the same principle. Any software needs to [18]:

- Fetch, execute, save instructions to the microprocessor
- Allocate memory for storing data in random access memory
- Cache data in CPU

There are many optimization methods and Matlab provides documentation on how to perform it. If someone asked the question what the best optimization technique is, there would probably be no answer. Matlab recommends a procedure of optimization of functions implemented in that language, but it does not specify what to do with the algorithm itself, as there is infinitely many of them. It needs mentioning that the best programmers seen by the author of the thesis developed some kind of intuition as they have seen and worked with plenty of similar problems instead. Their solutions are very often very creative and combine many techniques mentioned in [4], [15], and many others. The mortal programmers however, like the author himself, need to systematically use the methods described in these publications to find out the most optimal way of re-engineering the algorithms.

Some work was already done to optimize the algorithm as it is possible to call it with preprocessed data. The algorithm will then be able to use the data passed as arguments and save the time which otherwise would be spent on computations. For the details, the reader is encouraged to study the `vararg` argument, which in many cases contains the preprocessed information.

The algorithm was further improved. The most important improvement for this thesis was based on the observation that Matlab provides methods which are universal for many applications, and many operations can be simplified or rewritten

so that the domain of the functions becomes smaller but the speed of computation increases. So was the case in the `findn` optimization. There were three versions of the `findn` function.

The applied solution is based on an interesting observation by [18] and [15]. The optimized version of the `findn` function and the context the functions is running in performs one large memory allocation instead of multiple allocations. The original version of the code contains a call to `findn` function without passing the information about the dimensionality of the data. For this project, all the data will always be two or three-dimensional. Therefore the modification in the thesis removes the reference to the original `findn` function, which is designed to work in *n*-dimensions and instead the information about the size of the dimensionality is passed by creating a three-dimensional matrix `coord`.

The preliminary listing of the use of the `findn` function was:

```
coord=findn(im3D3);
```

The code was replaced with:

```
coord=zeros(nnz(im3D3),3);
[coord(:,1),coord(:,2),coord(:,3)]=ind2sub(size(im3D3), find(im3D3));
```

The methodology of this optimization is common for many projects as most computer programs need to use multidimensional arrays. The code is designed to execute the function `ind2sub`, but this observation applies to many functions, and not only this one. The execution of the function `ind2sub` is preceded by a call to `zeros`, which allocates random access memory for the result. When the function is called, there are always two functions: the caller and the callee, where the callee is the function being called. The callee in this case does not need to perform any memory management operations, as memory has already been prepared beforehand and therefore multiple memory operations, which would be performed by the callee do not occur. This finally brings the optimization in time of execution.

The fastest solution for any problem is to precalculate the output for every possible input in the domain of the function and then saving the precomputation result. When the function is queried that would just require finding the precomputed result and returning it. This would create a solution which is applicable for the commercial market as long as the database of all possible computations would have a size which can be stored on a modern server. The problem with this solution would be the computational time to generate all the possible solutions and the limited storage space.

The code was optimized and the single call for the function showed the results visible in the results section of this report. One must however be aware of the elephant in the room - the fact that the Genetic Algorithm is dependent on a random number generator. There are three important issues connected to it.

- Some runs of the GA will end up faster because the mutations happened to be more fortunate
- The GA can be made deterministic for testing purposes by limiting the maximal number of generations
- The projection time can be measured separately from GA.

The three points show the reason why the results in runtimes of consecutive cadaver reconstructions vary on lot and it is not possible to base the performance comparison solely on the difference between the reconstructions.

Although it is virtually not possible to run the same reconstruction twice and obtain exactly the same timing, there are methods to measure just the deterministic part of the code.

The algorithm produces different results for each run as the Genetic Algorithm is non-deterministic, but rather based on random decision.

There are alternatives available to the Genetic Algorithm. All of the alternatives aim at finding numerical solution to the given problem. Matlab provides a toolbox called Global Optimization Toolbox. This toolbox contains multiple functions for solving optimization problems. Some of the alternatives, according to Matlab's documentation [7] are:

- Genetic Algorithm
- Particle Swarm
- Surrogate Optimization
- Simulated Annealing
- Multiobjective Optimization

Continued research on Matlab documentation indicates that many of these methods might be of interest for further research. Particularly surrogate optimization might be interesting as it can accept constraints on the cost function. Choosing method which reduces the random variation in run time might be of great interest because it would allow to easily construct an accurate benchmark for the entire reconstruction. According to the results from this study, the run time for the full reconstruction is unpredictable with the current method.

## 6.3 Recompilation of Matlab code to C++

The recompilation and generation of C++ code was less effective than the refactorization as not all Matlab code can be automatically translated to C++ and the developer needs to re-implement some of the functions in C++. The Matlab function

`findn` and the `Cell` data structure in Matlab could not be automatically translated to C++ and these functions and data types were reimplemented. As Matlab provides the most effective implementation of the algorithm, it is difficult to re-create the most optimal version of the algorithm using pure C/C++.

Integration of the source code written in C++ with Matlab created some issues for stability of the application. C++ does not have memory management [18] [15] and requires the developer to handle memory management manually. This is an important disadvantage of developing code aimed to run inside another process. A memory allocation error may destabilize Matlab and the program will ask for restart as its memory becomes corrupted. For an implementation running on a cluster, a watchdog needs to be written in order to notify operators about possible memory errors for any C implementation. No permission was granted to publish the source code used in the research, therefore no discussion around it is presented.

## 6.4 Ethical discussion

The cadavers used in the research originate from the research performed by [19]. These samples consisted of proximal cadaver femurs collected at Kuopio University Hospital. Ethical approval for the collection of samples was granted by the National Authority for Medicolegal Affairs with permission number 5783/04/044/07. The output from this research may be used in clinical research to improve the speed and quality of digitally reconstructed radiographs.

The thesis were written with awareness of ethical issues. One of them is the data protection issue. For every bit of data stored there is a danger of leakage of the data to the third parties which by no means should be allowed to access the research data. The third party may be able to use the data to harm and not for the benefit of the patient. The data collected for the research will be available for the future generations and no one can be sure that it will be used for the benefit of the patient. Therefore sensitive data should always be protected by disk encryption and only trusted parties should be allowed to access the data.

Yet another concern about the ethical issues in this publication is the inability to tell how much of the data used for the research should be shared with the third parties as it may be necessary to share it in order to continue the research. The third party which receives the data must be a trustworthy organization, for example the University.

A natural, but not directly connected to the research issue is which patients should receive the treatment first. While the solution might help some patients, the amount of resources for the treatment is always limited. Therefore the researchers should always strive to create the effective solution.

## 6.5 Dimensionality and the reduction of the datapoints

The theory section is started with claim that the reduction of the data set can be improved to increase the speed of computations. This was not necessary because a different solution with many advantages was developed during this thesis. The solution discussed in this thesis does not lose accuracy while performing the projection. While this is a large advantage of this solution and shows that the optmization may be done in the clean way, it may be insufficient for some applications where the reduction of size of the data is necessary, for example due to limited hardware resources in clinical environments. The methodology chosen for this project (refactorization and reimplementation of the Matlab code) turned out some impressive results.

## 6.6 Conclusion

Parts of the algorithm were optimized and it was shown that the optimization indeed brings an increase in the performance, but due to the non-deterministic nature of the Genetic Algorithm the optimization is not visible when running the reconstructions.

## 6.7 Further improvements

The following is a list of potential improvements to the project:

- The most time-consuming functions are listed in the results section. Further analysis is required to find out which functions need more optimization and what effort is required to optimize them.
- In the current implementation the projector works with tetrahedrons. For each tetrahedron the average BMD is computed during the calculation. This can be improved as the data is extrapolated and multiple data structures contain redundant information. It would be possible to extrapolate the information on fly without creating the need to refer to the random access memory. This could possibly increase the speed of the solution.
- Although the method used to measure the performance of the algorithms shows how much improvement was made, it is not the only method of measuring speedup of an application. The other methods which can be used for measurements of the performance of algorithm might be counting CPU instructions. The method of counting the CPU instructions rather than the time it takes to measure the process has some advantages over measuring the time. This method is described in [11] [12]. The solution is possible and relatively easy to implement. The implementation would be based on running the code on a modified version of CPU emulator and the modified version would save

the amount of CPU instructions in a file when the execution of the code is done.

- Other implementations of DRR are available, like the Attenuation Fields, Attenuation Box or Splatting. It would be possible to implement some parts of the available DRR programs, extract the projection modules and apply them to this solution. These solutions may use different data structures than the structures used in this solution. It may be necessary to translate these structures to structures compatible with the solution used in the thesis to make the program run. The translation process may itself be time-consuming for the CPU.
- The image from the DXA object contains landmarks which need to be placed manually. An algorithm can be developed to automate the landmark placement.

# Bibliography


[1] C. Cooper, L. J. Melton. “Epidemiology of osteoporosis”. *Osteoporosis international 2, S2– S8* (1999).

[2] W. Cai and G. Sakas. “Maximum intensity projection using splatting in sheared object space”. *Computer Graphics Forum* (1998).

[3] “Consensus development conference: prophylaxis and treatment of osteoporosis”. *Osteoporosis international 1, 114–117* (1991).

[4] T. H. Cormen, (Ed.). *Introduction to algorithms*. 3rd ed. MIT Press, Cambridge, Mass, 2009. 1292 pp.

[5] A. Cranney, S. A. Jamal, J. F. Tsang, R. G. Josse, and W. D. Leslie. “Low bone mineral density and fracture burden in postmenopausal women”. *CMAJ: Canadian Medical Association journal = journal de l’Association medicale canadienne* (11, 2007).

[6] S. Ghafurian, D. N. Metaxas, V. Tan, and K. Li. “Fast generation of digitally reconstructed radiograph through an efficient preprocessing of ray attenuation values”. In: *Medical Imaging 2016: Image-Guided Procedures, Robotic Interventions, and Modeling*. Medical Imaging 2016: Image-Guided Procedures, Robotic Interventions, and Modeling. International Society for Optics and Photonics, 18, 2016.

[7] T. M. Inc. *Matlab Version 9.9.0 (R2020b)*. 2021st ed. The MathWorks Inc., Natick, Massachusetts, 2021.

[8] J. A. Kanis and O. Johnell. “Requirements for DXA for the management of osteoporosis in europe”. *Osteoporosis international 16, 229–238 (2005)* (2005).

[9] J.-Y. Reginster and N. Burlet. “Osteoporosis: a still increasing prevalence”. *Bone 38, S4–S9* (2006).

[10] I. T. Jolliffe. “PRINCIPAL COMPONENT ANALYSIS: a BEGINNER’s GUIDE - i. introduction and application”. *Weather* (1990).

[11] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood. “Pin: building customized program analysis tools with dynamic instrumentation”. *ACM SIGPLAN Notices* (12, 2005).

[12] N. Nethercote and J. Seward. “Valgrind: a framework for heavyweight dynamic binary instrumentation”. In: *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation - PLDI ’07*. the 2007 ACM SIGPLAN conference. ACM Press, San Diego, California, USA, 2007.

[13] D. Russakoff, T. Rohlfing, K. Mori, D. Rueckert, A. Ho, J. Adler, and C. Maurer. “Fast generation of digitally reconstructed radiographs using attenuation fields with application to 2d-3d image registration”. *IEEE Transactions on Medical Imaging* (2005).

[14] N. Sarkalkan, H. Weinans, and A. A. Zadpoor. “Statistical shape and appearance models of bones”. *Bone* (2014).

[15] J. Skeppstedt and C. Söderberg. *Writing Efficient C Code: A Thorough Introduction*. 1st edition. Skeppberg, Flyinge, 15, 2011. 784 pp.

[16] R. S. Snell. *Clinical Anatomy by Regions*. 9th ed. Lippincott Williams & Wilkins, a Wolters Kluwer business, 2012.

[17] K. L. Stone, D. G. Seeley, L.-Y. Lui, J. A. Cauley, K. Ensrud, W. S. Browner, M. C. Nevitt, S. R. Cummings, and Osteoporotic Fractures Research Group. “BMD at multiple sites and risk of fracture of multiple types: long-term results from the study of osteoporotic fractures”. *Journal of Bone and Mineral Research: The Official Journal of the American Society for Bone and Mineral Research* (2003).

[18] A. Tanenbaum and H. Bos. *Modern Operating Systems*. 4th edition. Pearson, Boston, 10, 2014. 1136 pp.

[19] S. Väänänen. “Functional imaging of proximal femur by combining dual energy x-ray imaging and finite element simulations” (2014).

[20] S. P. Väänänen, L. Grassi, G. Flivik, J. S. Jurvelin, and H. Isaksson. “Generation of 3d shape, density, cortical thickness and finite element mesh of proximal femur from a DXA image”. *Medical Image Analysis* (2015).

[21] Wenli Cai and G. Sakas. “DRR volume rendering using splatting in shear-warp context”. In: *2000 IEEE Nuclear Science Symposium. Conference Record (Cat. No.00CH37149)*. 2000 IEEE Nuclear Science Symposium. Conference Record. IEEE, Lyon, France, 2000.

[22] D. Whitley. “A genetic algorithm tutorial”. *Statistics and Computing* (1994).

[23] WHO. “Assessment of fracture risk and its application to screening for postmenopausal osteoporosis. report of a WHO study group”. *WHO Technical Report*. 1998th ser. (1999).