UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Comparison of iterative and statistical cone-beam CT reconstruction methods

Dissertation presented by
Alexandre ARNOULD

for obtaining the Master's degree in
Electrical Engineering

Supervisors
John LEE, Benoît MACQ

Readers
Arnould GEELHAND DE MERXEM, Rudi LABARBE , Merence SIBOMANA

Academic year 2017-2018

Acknowledgements

The work behind this thesis would not have been possible without the support and the valuable advice of my supervisors, Professor Benoît Macq and Professor John LEE. They guided me through the relevant resources in order to solve the problems I encountered.

Another key person behind the many discussions and decisions detailed in this document is Arnould Geelhand de Merxem. He is a UserMedia researcher. His explanations and help have been more than valuable from the start until the end.

Finally, I would like to thank my parents and friends for proofreading the whole document and for supporting me throughout its redaction.

Abstract

The goal of this master thesis has been to compare iterative and statistical algorithms in the context of reconstruction methods for Cone Beam Computed Tomographies (CBCT). These families of algorithms are more efficient with limited data and more robust against artifacts in comparison with the standard reconstruction technique: FDK.

The paper starts with an introduction to the context and the use of CBCTs, with special attention to the applications within protontherapy. The next section presents the reconstruction algorithms that could be used with CBCTs: filter back projection algorithms, algebraic reconstruction techniques, TV-based algorithms and statistical algorithms. The third section deals with the method used to produce and asses the reconstructions. It also introduces some image quality metrics. A detailed explanation of the TIGRE toolbox is also part of that section. Finally the fourth and fifth sections are focusing on the results produced by the tested reconstruction algorithms and the discussion of these results. The discussion emphasises the comparison of the algorithms with FDK and discusses whether or not they could be a realistic alternative.

Contents

Chapter 1

Introduction

1.1 Protontherapy [1]

Cancer is the second leading cause of death worldwide (see Figure 1.1). Moreover, it is still growing on a yearly basis as shown on Figure 1.2. The global trend is that the number of patients with cancer is rising significantly. Although we can see that the group of people over 50 years of age (red and grey on Figure 1.2) are more subjected to the rise in number of cancers, people below 49 years of age (blue and green on Figure 1.2) are also affected by the increase of cancers.

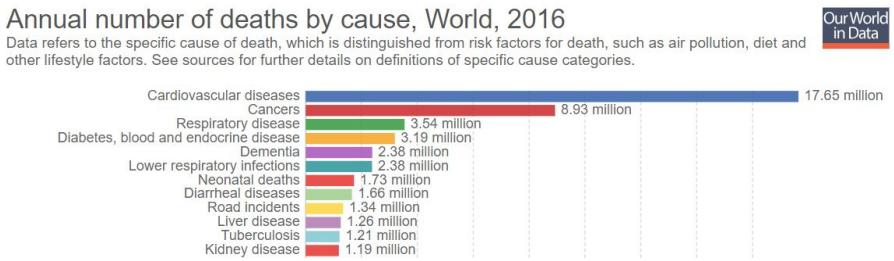


Figure 1.1: Number of deaths by cause [24]

Due to the stronger rise of the last twenty years and the technological advancements, the chances of being cured from cancer have improved. For example, in the UK, cancer survival chances have doubled in the past 10 years [25]. The classic way to cure cancer is through radiotherapy or surgery if it is discovered in an early stage. Otherwise, if already advanced on a local basis, chemotherapy is coupled with radiotherapy. Radiotherapy consists of delivering photon beams to the patient, affected with cancer, to kill the malignant cells.

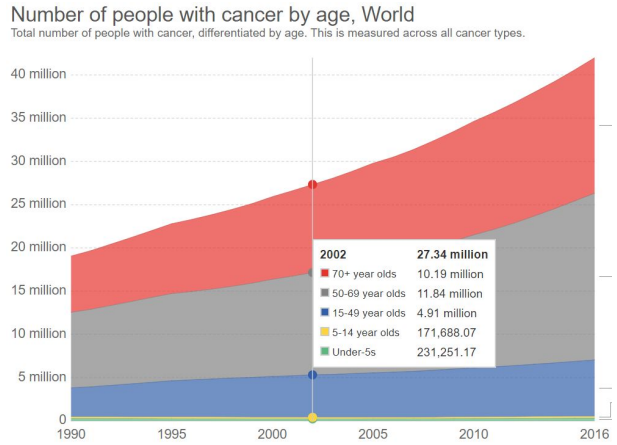


Figure 1.2: Evolution of number of people with cancer[23]

The increase in number of people diagnosed with cancer has encouraged research to look to other possible solutions [15]: vaccines, immunotherapy and even nanoparticles. Another cure that has been used for treating cancer is protontherapy.

Protontherapy is similar to radiotherapy as it also delivers beams of high energy to the patient. It is however different by the particles which compose the beam. As radiotherapy uses photons, protontherapy, as the name suggests, uses protons.

Protontherapy is interesting because it is tackling one of the main drawbacks of radiotherapy. Radiation therapy has poor local control. This means that not only the affected organs and tissues, but also upstream and downstream organs and tissues are submitted to the prescribed dose. By using protons, protontherapy does not have that problem. The difference between photons and protons lies with the dose deposited inside the body. As particles such as protons and photons enter the body and interact with the matter, the dose they bear is deposited in the body. Figure 1.3 shows the comparison between the deposition of the dose by photons and protons. As shown by the dotted line, the photon dose is deposited in a linear way throughout the body. In contrast, protons have a dose deposition that has a specific shape: a Bragg peak (the full line in Figure 1.3), where can be seen how, after travelling some distance through the body, the proton deposits most of its energy.

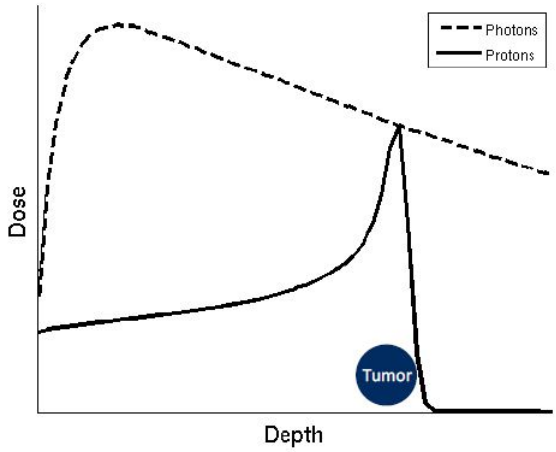


Figure 1.3: Proton penetration:Bragg-peak[1]

The main advantage of such a Bragg peak is an improved local control: physicians and MDs can model the dose control and target a certain volume with multiple Bragg Peaks (Spread Out Bragg Peaks). This advantage goes with a drawback: one has to target the tumour in a really precise way. The dose at the tumour is significantly higher comparing to the dose exposed to the surrounding tissues. This implies that the Bragg Peak needs to be precise enough to avoid damaging healthy tissue and to target the right volume precisely so as to get rid of all cancerous cells.

1.2 Cone Beam CT

The previous section explained why precision is the key to a successful protontherapy. This means that physicians and doctors need to have the most accurate possible information about the tumour and the healthy tissues surrounding the tumour, so that the correct dose can be prescribed and optimised for the patient. To achieve this, imaging is needed during cancer treatment with protontherapy (as it is for all other methods of curing cancer). When looking at the imaging in the protontherapy process, there are three stages:

1. Pre-treatment
2. Per-treatment
3. Post-treatment

Pre-treatment [1] is purely for treatment planning and happens before protontherapy: doctors need a first, complete and precise view of the tumour and its surroundings in order to plan the right protontherapy. This stage is often performed out-room, i.e. not in the same room as the room where the protontherapy will be performed. As precision and completeness are needed, the preferred imaging tool here is an MRI (Magnetic Resonance Imaging) or a PET scan (Positron Emission Tomography).

As protontherapy needs great local control, imaging is also performed right before the protontherapy. This is called **per-treatment** imaging [1]. Every change in the tumour size or location, every change in the surroundings may alter the quality of the treatment if it is not considered properly. So, for per-treatment imaging, CBCT (Cone Beam Computed Tomography) is used for positioning and setting-up the patient. In addition, other types of imaging are used for range verification and plan adaptation.

Lastly, **post-treatment** is for the patient follow up: making sure the tumour has been eliminated and taken care of.

As mentioned above, a CBCT is, in case of protontherapy, used in per-treatment. IBA merged the equipment for the CBCT and protontherapy, to allow to combine both. This can be seen on Figure 1.4 which shows IBA's Proteus One: one device with a CBCT, a patient's positioner and an Intensity Modulated Proton Therapy.

Figure 1.4: Proteus one [18]

As the name suggests, CBCTs use cone beams instead of a (thick) fan beam. There are some other differences that can be noticed on Figure 1.5: a CBCT needs a planar detector while a modern CT needs a linear arced detector. This implies that in the case of a CT there is a helical movement between the source/detector and the patient[1]. However, the main principles of CTs and CBCTs are identical: x-rays are sent through the body and detected by a detector which measures the photons which have been attenuated by interaction of the photons with the body. This measured intensity is given by the Beer-Lambert Law [1]: $I(s) = I(0) * e^{-\int_0^s \mu(\eta)d\eta}$. $I(0)$ is the incident intensity, s the distance traveled through the object and μ the attenuation coefficient.

This equation is for monochromatic x-rays, in the real case the x-rays are polychromatic. (In Section 4.2 the used projections are created from monochromatic x-rays).

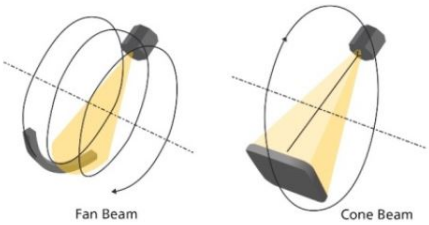


Figure 1.5: CT (left) and CBCT (right) [1]

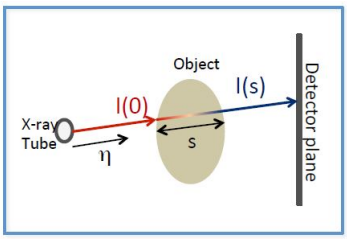


Figure 1.6: CT scheme [1]

After having received the projections on the detector the next step is the reconstruction: finding the attenuation coefficient μ for each voxel. With the μ of each voxel the object can be reconstructed. Figures 1.7 and 1.8 show the difference between a projection and a reconstructed image. To go from projections to reconstructions algorithms are needed.

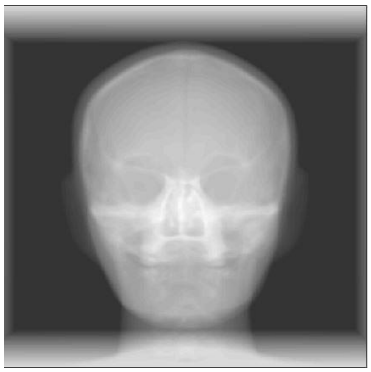


Figure 1.7: Projection of a head

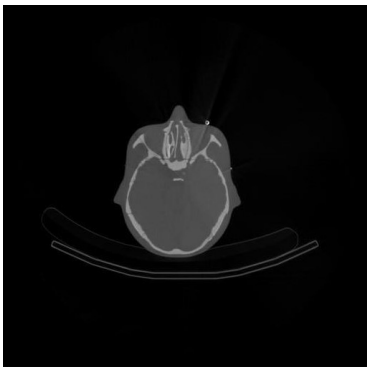


Figure 1.8: Slice of the head after reconstruction

CBCT also has other purposes outside protontherapy. Its largest use is in dentistry as shown on Figure 1.9. But the CBCTs used here are a lot smaller, and thus not completely similar to the CBCT used in the protontherapy context.

Figure 1.9: CBCT scanning in dentistery [16]

Other applications are, for example, orthopedics or implantology (planning surgical implants)

[26]. At the Université Catholique de Louvain, the advantages of CBCT, speed and cheaper cost, are used for a social purpose. The searchers of ImageXEco, a spin-off of the UCL, are looking at the CBCT aiming at making CT technology affordable for Third World Countries. Time savings and cost reductions go with a lowered image quality and a higher inclination to artifacts. This is the trade-off which is made when using cone beams.

1.3 Artifacts in CBCT

As mentioned in the section above, CBCT trades off image quality for time gain and cost savings. This can be seen in the image quality, which will be discussed later on. The trade-off is most visible through artifacts. This section will discuss some of them.

1.3.1 Compton scattering

Scattering refers to the fact that a photon is scattered by an electron when passing through matter (in the case of CBCT, the patient's body). There are two sorts of scattering: coherent [21] and compton [22] scattering. The difference lies within the energy carried by the photon. In coherent scattering there is insufficient energy for a proper energy transfer to occur and so, the photon is only deflected. In compton scattering however, the photon is not only scattered but also loses energy. This is the most prominent artifact that is present in CBCTs.

The deflection and the energy attenuation degrade the image quality as the detected photon is not of the right energy and is not detected at the right place.

As CBCTs have a planar detector, they do not need the source and the detector to change the linear position with respect to the patient, as is the case for a CT (on Figure 1.5 it is visible that the CBCT only rotates around the patient while, in case of a CT, the detector and source also translate along the patient). However, this means that a CBCT is more subjected to scatter since the scatter can occur in two directions, which is not the case for the CT which only has scatter along one dimension.

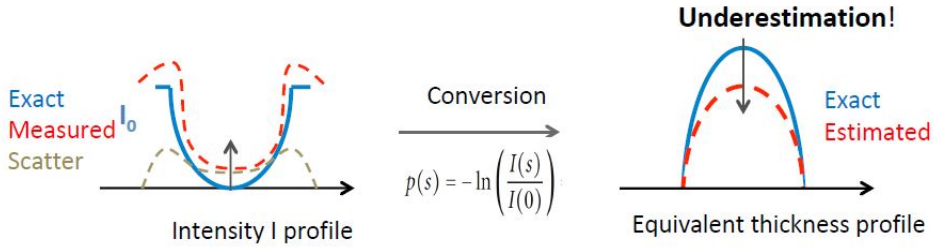


Figure 1.10: Scatter Radiation [1]

On Figure 1.10, the consequences of scattering are shown: the measured radiation is higher than the exact radiation, due to the scattered photons. So, there is a need for conversion and underestimation of measured projections.

1.3.2 Beam Hardening [10]

Another artifact is beam hardening. It is similar to scattering as it has the same consequences: more photons are to be detected, which leads to dark streaks on the image along the line of the greatest attenuation.

Beam hardening refers to the fact that low energy photons are more subjected to attenuation than high energy photons, resulting in a harder beam: a beam with a higher energy.

1.3.3 Other artifacts

Besides the two artifacts discussed before there are also other artifacts:

- **Sampling:** aliasing due to angular and spatial sampling
- **Geometric artifacts:** cone beam approximations
- **Noise:** statistical error in photon count [10]

Chapter 2

State of the Art

There are several possible reconstruction techniques for CBCTs, but not all are implemented within the medical world. FDK is the standard reconstruction technique. Although FDK is very fast, it struggles with artifacts and noise. Certainly in cases where the data is not idealistic or that there is no sufficient data. Consequently, alternative iterative and statistical algorithms have been implemented as they are known to produce better results with non idealistic data.

This section will introduce different existing algorithms for image reconstruction in the case of CBCTs [13]. The algorithms can be split up in different families: filtered back projection algorithms, algebraic reconstruction techniques (the SART-family and the krylov subspace based algorithms), the TV (Total Variation) algorithms and the statistical algorithms. This division is based on the classification used within the TIGRE library [13].

We could split the families into two main groups [13]. First, FDK (Feldkamp-Davis-Kress) solves the inverse radon transform. The radon transform being a line integral, as shown on Figure 2.1.

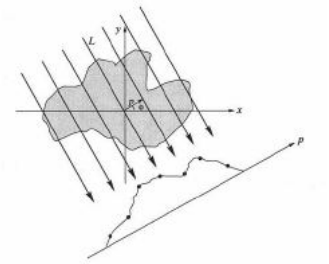


Figure 2.1: 2D-radon transform [2]

The other group, the iterative algorithms, minimizes the following equation:

$$\hat{x} = argmin_x ||b - Ax||^2 + G(x) \tag{2.1}$$

Where x is the image, b the projections, A a matrix representing the intersections between the x-rays of the source and the voxels of the object and G being a regularisation function, but that is not used in all of the algorithms.

The formulation in equation 2.1 gives the possibility to use advanced algebraic resolution techniques. But the size of the matrix A is the challenge, almost all iterative reconstruction techniques use at least one Ax multiplication and one $A^T b$, which are very time consuming. This is the main reason FDK is still more popular even though the image quality is less. The image quality is much improved in the iterative and statistical algorithms due to the use of a priori within the reconstruction.

2.1 Filtered back projection algorithms [12] [4]

The best known algorithm of this family, with respect to CBCTs, is the FDK algorithm. This is the only non-iterative algorithm family. Nowadays, image reconstruction for CBCTs is mainly handled by the FDK algorithm.

This is remarkable knowing the fact that iterative algorithms outperform the FDK algorithm which concerns image quality, as will be discussed in Sections 4 and 5. FDK needs nearly perfect data and perfect conditions which are not always possible in real cases: projections that are not subjected to a lot of noise, projection angles that cover more than 180 degrees, etc. When the conditions are not ideal artifacts are influencing the reconstructed image (beam hardening, scattering, ...).

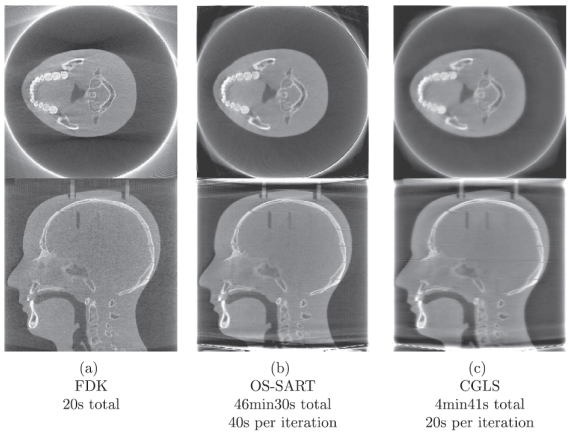


Figure 2.2: Comparison of FDK with other algorithms: execution time and image quality [13]

Still FDK is the most used and this is due to its speed. Figure 2.2 illustrates the advantages and drawbacks as the FDK algorithm is a lot faster in execution time but does have a more noisy image than the two other reconstructed images. This will become more clear in Sections 4 and 5 where the performances of all algorithms are diplayed and discussed.

The algorithm starts using the projections:

$$g(\lambda, u, v) = \int_0^{+\infty} f(\vec{a}(\lambda) + t\vec{\beta}(\lambda, u, v))dt$$

where u, v are the coordinates on the detector (flat pannel), λ is the angle, $\vec{a}$ is the x-ray source and $\vec{\beta}$ is the vector connecting source and detector. D is the distance (mathematical definition) between the detector and the source. In the equations below it is represented by R.

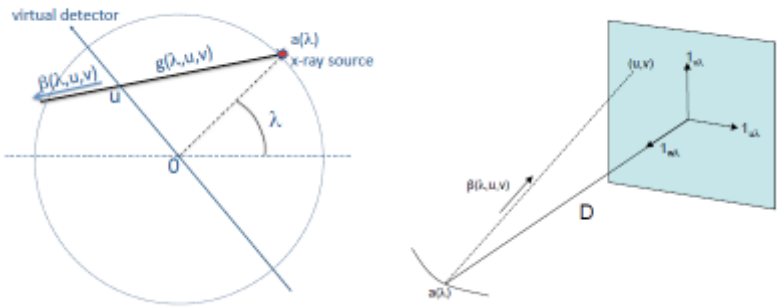


Figure 2.3: Geometry of the CBCT [12]

The FDK algorithm consists of three steps:

1. Add a weighting to the different projections ($g(\lambda, u, v)$): $R/\sqrt{R^2+u^2+v^2}$
2. Computing a convolution between the weighted projections and a ramp filter $g^F(\lambda, u, v)$
3. Back projection: $f(\vec{x}) = \frac{1}{2}\int_0^{2\pi} \frac{R^2}{(R - xcos(\lambda) - ysin(\lambda))^2} g^F(\lambda, U(\vec{x}, \lambda), V(\vec{x}, \lambda))d\lambda$

with $U(\vec{x}, \lambda) = \frac{R(-xsin(\lambda) + ycos(\lambda))}{R - xsin(\lambda) - ycos(\lambda)}$ and $V(\vec{x}, \lambda) = \frac{Rz}{R - xsin(\lambda) - ycos(\lambda)}$

2.2 Algebraic reconstruction algorithms

2.2.1 SART-type family [14] [3]

SART stands for Simultaneous Algebraic Reconstruction technique. It is an improvement of the ART (Algebraic Reconstruction Technique) algorithm. The principle of the ART algorithm is shown on Figure 2.4. Each square is a voxel (3D pixel) and the algorithm computes μ, the attenuation coefficient. So ART solves iteratively the equation system given by $Ax = b$ (based on projections from different angles) to find the attenuation factor of each voxel. So in the case on N^2 voxels the number of equations extracted from the projections needs to be larger (or equal) than N^2.

The total attenuation of each ray can be measured on the flat panel. Then this total attenuation is divided on the different voxels. The next step is using the other projections (with a different angle) to have another equation (as shown on Figure 2.4: the first iteration is the horizontal projection and the second one, the vertical projection). The second iteration is used in the following way: due to the first step we have a first estimation of the μ of each voxel. Then compare the sum of the estimated attenuations with the actual one that has been measured through another projection. This difference is divided through the different concerned voxels. After multiple iterations the voxels will have (nearly) their accurate attenuation factor, μ.

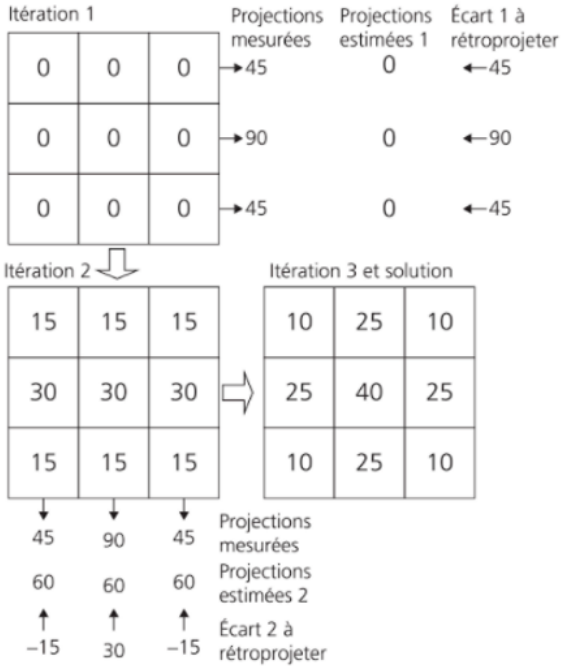


Figure 2.4: ART algorithm [14]

Being an improvement of ART, SART is a superior implementation due to the fact that there is a simultaneous application of the error correction terms as computed by ART for all rays in a given projection. ART is a sequential method, which means that the correction is applied to a single ray-sum and then passes on to the next equation. This sequential approach is not the case in the SART method as it works projection by projection and not row by row. Moreover there is a longitudinal weighting of the correction terms back-distributed along the rays. Lastly, it is using bilinear elements for discrete approximation to the ray integrals of a continuous image.

The SART-family consists of different algorithms making the trade-off between speed and accuracy of the solution. The family is based on the following iterative equation:

$$x^{k+1} = x^k + \lambda^k V A^T W(b - Ax^k) \tag{2.2}$$

With x the reconstruction, λ a constant (by default 1), b the projections, A a matrix representing the intersections between the x-rays of the source and the voxels of the object, and V and W weighting matrices.

SIRT, OS-SART and SART are three different algorithms, SIRT stands for Simultaneous Iterative Reconstruction Technique and OS-SART for Ordered Subset (OS) SART. The difference is within the number of projections used for each update: all projections, subset of projections or projection by projection. This implies that, the more data used in one update will diminish the time per iteration but increase the number of iterations needed to converge towards the wanted solution.

2.2.2 Krylov-subspace based algorithms [13] [11]

In comparison with the SART-family algorithms the Krylov-subspace based algorithms, are faster to resolve the linear equations. The algorithm representing this family within the TIGRE toolbox is the CGLS (Conjugate Gradient Least Squares) algorithm.

Due to the fact that it goes faster to resolve the equations it has the same results in ten times less iterations than the SIRT algorithm, for example.

The higher speed is due to the fact that it iterates on the Krylov Subspaces. These are formed by the k first powers of A on b: $K_r(A, b) = span\{b, Ab, A^2b, ..., A^kb\}$. This has the main advantage not to work with matrix (A) but with vectors (Ab, A^2b, ...), as in the case of CBCTs the size of the A matrix is very large and not easy to store on an average computer. Looking at the equation $Ax = b$ and the least squares problem $min_x||b - Ax||^2$ the computational challenge lies within the size of A. For example, A being a M by N matrix with M and N respectively the number of scanned angles times the amount of detectors and N the number of voxels within the image the size of A can fast get out of proportion. For a 360 degrees scan of an image of size 512x512 with a detector of size 512 we have a matrix of $512 * 360$ by 512^2 .

The size of A causes a memory problem. The proposed solution by the CGLS algorithm is to only use the needed information and not the whole matrix. It only requires one vector-matrix multiplication with A and A^T per iteration. To do the vector-matrix multiplication the matrix A is subdivided into blocks. This leads to blockwise vector-matrix multiplication and permits to use and acces the whole matrix A, which would not be possible otherwise. This is shown in the equation below with M by N matrix A and vector v:

$$Av = \begin{bmatrix} A_1 v \\ A_2 v \\ . \\ . \\ . \\ A_n v \end{bmatrix} \tag{2.3}$$

Each A_j contains data from the projections and is of size M/n by N and v is of size N by 1.

That way the blocks of matrix A and A^T can be loaded only when needed in the matrix-vector multiplication.

2.3 TV-algorithms [13] [6]

This technique is a bit different from the others because TV-regularization (TV stands for Total Variation) is often used for denoising images from undersampled data. So it is a post-processing algorithm (for example in compressed sensing [8]). It is based on the TV-norm: $||x||_{TV} = \sum_{i,j,k} ||\nabla x[i,j,k]||_{l2}$ [8].

But in the case of ASD-POCS (Adaptive-Steepest-Descent Projections On Convex Sets), the TV-norm minimisation is part of the algorithm. As the name implies it contains two parts: a projection on convex sets part and an adaptive-steepest descent part. POCS means that you use two convex sets and look for their intersection to find the wanted solution. A convex set being a set containing the interval $[a, b]$ for all a and b elements of the set [19].

Figure 2.5 shows how ASD-POCS works. I(ϵ) consists of all images within tolerance ϵ. f_0 is the starting point and is adapted by dp due to POCS and dg due to the TV-minimization, which finishes in f^*. The main advantage of TV-norm minimisation with respect to conjugate gradient or steepest descent is the fact that it handles incomplete data in a much better way. While the standard techniques will easily find the image that minimises the data fidelity term for complete data sets, it will be less efficient with incomplete data sets.

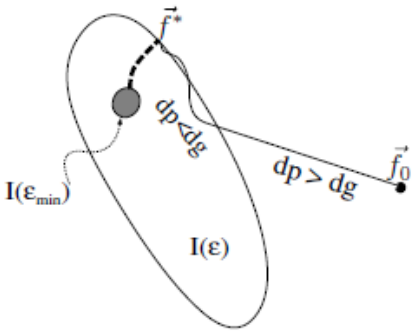


Figure 2.5: ASD-POCS algorithm [6]

The main drawback of TV-minimization based algorithms is the fact that there is no shading and that it gives a too simple reconstruction without enough details (cartoon-like image). As their advantage is that they perform much better than other algorithms in case of extremely noisy data as well as when the number of projections is reduced.

Other algorithms of the TV-regularization family have also been implemented within TIGRE for example SART-TV or B-POCS-TV-β. The TV-algorithm family has the specificity of combining algorithms with the TV-minimisation (mainly denoising the image). The minimization problem is expressed in equation 2.1 with the following regularisation function:

$$G(x) = ||x||_{TV} \tag{2.4}$$

2.4 Statistical algorithms [14]

This family of algorithms is one of the most recent in the field of CBCT. The one implemented in TIGRE is MLEM (Maximum likelihood Expectation Maximisation).

The way of working is similar to ART but where ART starts with a random initialisation (often 0), MLEM starts with an estimation and uses the weighting of the estimation to compute

the new values of the attenuation factors at each iteration. It then computes the ratio between the estimation of the total attenuation and the measured one. Then to compute the new estimation it goes as follows. You multiply the actual estimation with the sum of the two following ratios: the sum of the total attenuation of your column divided by the estimation with the total attenuation of the row divided by the estimation.

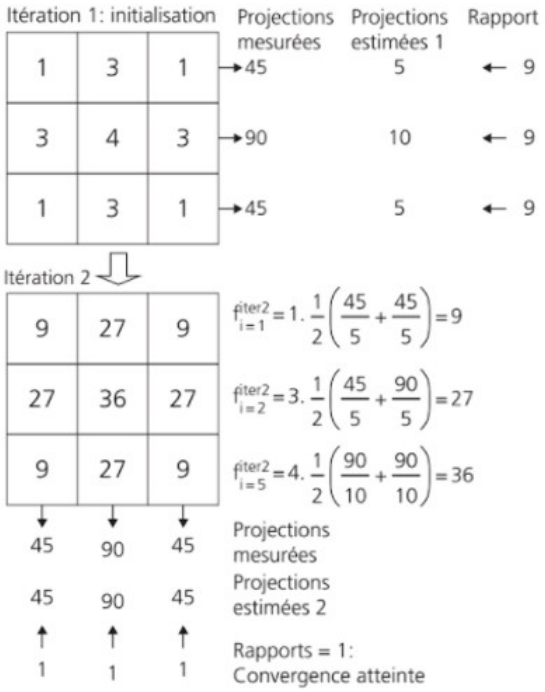


Figure 2.6: MLEM alorithm [14]

In pseudo code this is implemented as follows, for each iteration:

1. Compute the ratio y between the actual projections and the previous reconstruction that is projected: $b./Ax(x_k)$
2. Compute the following division: $z = Atb(y)./W$
3. $x_{k+1} = x. * z$

with b the projections, x_k the reconstruction, W a weighting matrix that is an initial reconstruction and Ax and Atb the projection and backward projection functions.

Chapter 3

Method

For the comparison of the results and the performances of the algorithms, the images need to be reconstructed. This chapter will give an overview of the used tools and method: the TIGRE toolbox, used for testing the algorithms, the workflow applied during the testing and the functions and script used within the workflow.

3.1 TIGRE [13]

The used algorithm library is an existing one: the TIGRE library. It has been developed by researchers from the University of Bath's Engineering Tomography Lab and CERN. To achieve the comparison of the algorithms three different parts of the toolbox were used: first the building blocks for the iterative algorithms, second the algorithms and third some other functions for testing those.

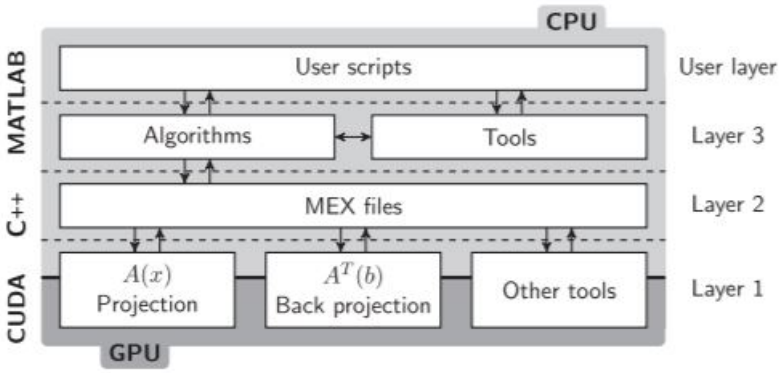


Figure 3.1: Structure of the TIGRE toolbox [13]

3.1.1 Building Blocks

When looking at reconstruction algorithms, there are two main building blocks that are needed: the projection part $A(x)$ and the back-projection $A^T(b)$ (represented in the TIGRE toolbox by the functions `Ax` and `Atb` respectively). Those two functions are at the bottom of the hierarchy as shown on Figure 3.1. As these two blocks are the most time consuming parts within the algorithms, they have been handled in GPU (Graphics Processing Unit) computing. They are utilised in the algorithms through the use of MEX functions, these make the connection between the low-level hardware of the GPU and the high-level language MATLAB.

The MATLAB functions Ax and Atb both need the same input arguments: a 3D matrix (the projections or the reconstructions), the wanted geometry and the angles of the projections.

3.1.2 Algorithms

Different reconstruction algorithms are implemented in the TIGRE toolbox. The different families have been discussed in Chapter 2. Here below is the list of all the algorithms represented in the toolbox:

- FDK
- Gradient descend family
  - SART
  - OS-SART
  - SIRT
- Krylov Subspace
  - CGLS
- Total Variation
  - ASD-POCS
  - OSC-POCS
  - B-ASD-POCS-β
  - SART-TV
  - PCSD
  - AwASD-POCS
  - AwPCSD
- Statistical Methods
  - MLEM

3.1.3 Other tools

The TIGRE toolbox features other tools as well.

First of all, image quality measure functions, such as RMSE (Root Mean Square Error) or SSIM (Structural Similarity), which will be discussed in a next Section (3.4). Next to these, there are functions for the display of 3D projections and also image processing tools such as functions for denoising the images.

Additionally there also exist functions to add noise to the projections, as the projection function `Ax` is simulating idealistic projections.

3.2 Workflow

Two data sets were used. Both are projections/reconstructions of the head.

One is a collection of projections in the format of HIS files with in addition an XML file containing the information concerning the projections.

The second is a series of reconstructions in the DICOM format, having its data split in two parts: one containing the upper part of the head, the other one containing the lower part.

3.2.1 Workflow starting from projections

Figure 3.2 shows in red the two entries: a folder with the projections in HIS format and an XML file containing the information of the projections: gantry angles, offsets, the projection number, etc.

The first step is to merge the HIS files in one large 3D-matrix. This is done by the MATLAB function `readHIS` shown in Appendix B.2.1. The output is a 3D-matrix containing all the projections that were within the HIS files. The HIS files have been obtained with an Elekta CBCT.

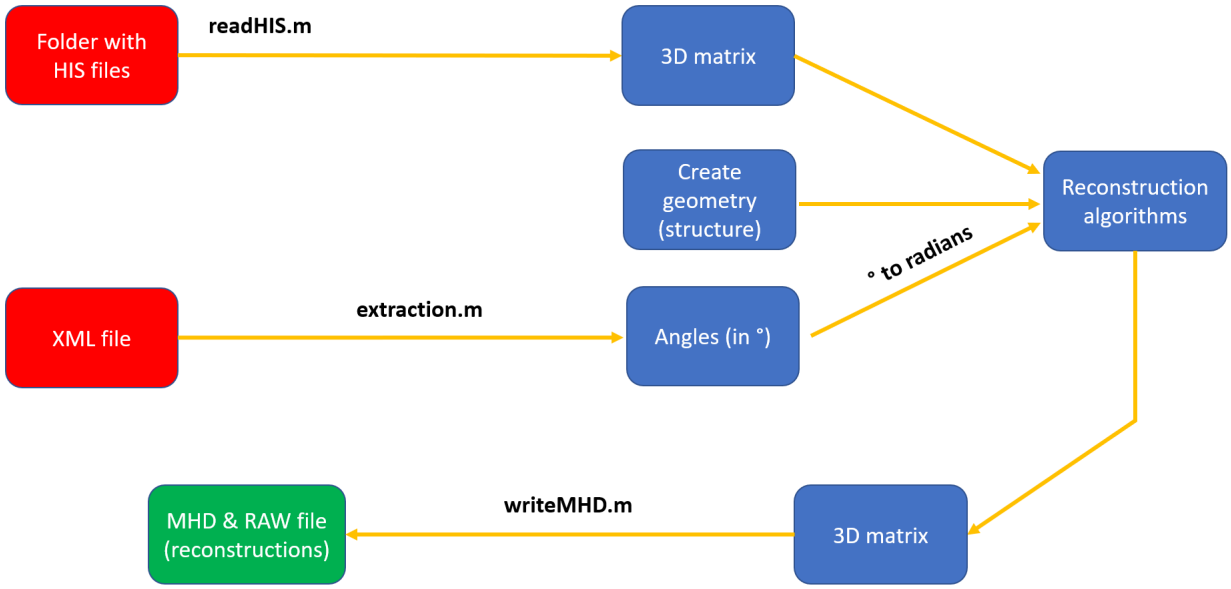


Figure 3.2: Workflow with HIS files

To reconstruct the head, starting from the projections, the projection angles are needed. These are contained in an XML file. The MATLAB function `extraction` in Appendix B.2.2 extracts the angles and the x and y offsets of each projection.

Next, a geometry need to be created. This is simply in the form of a MATLAB structure as shown in Appendix B.1. This structure will contain reconstruction parameters such as image size, voxel size, distance between the detector and the source, the distance between the source and the origin, etc.

Having the 3D-matrix, the geometry and the angles (transformed from degrees to radians), the reconstruction algorithms from the TIGRE toolbox can be used properly. The output of those algorithms is a 3D-matrix of the reconstructed body part. After that, this 3D-matrix is transformed into an MHD file that is linked with a RAW file. This makes it more generic and permits to also display the reconstructions with other tools, ITK-SNAP for example.

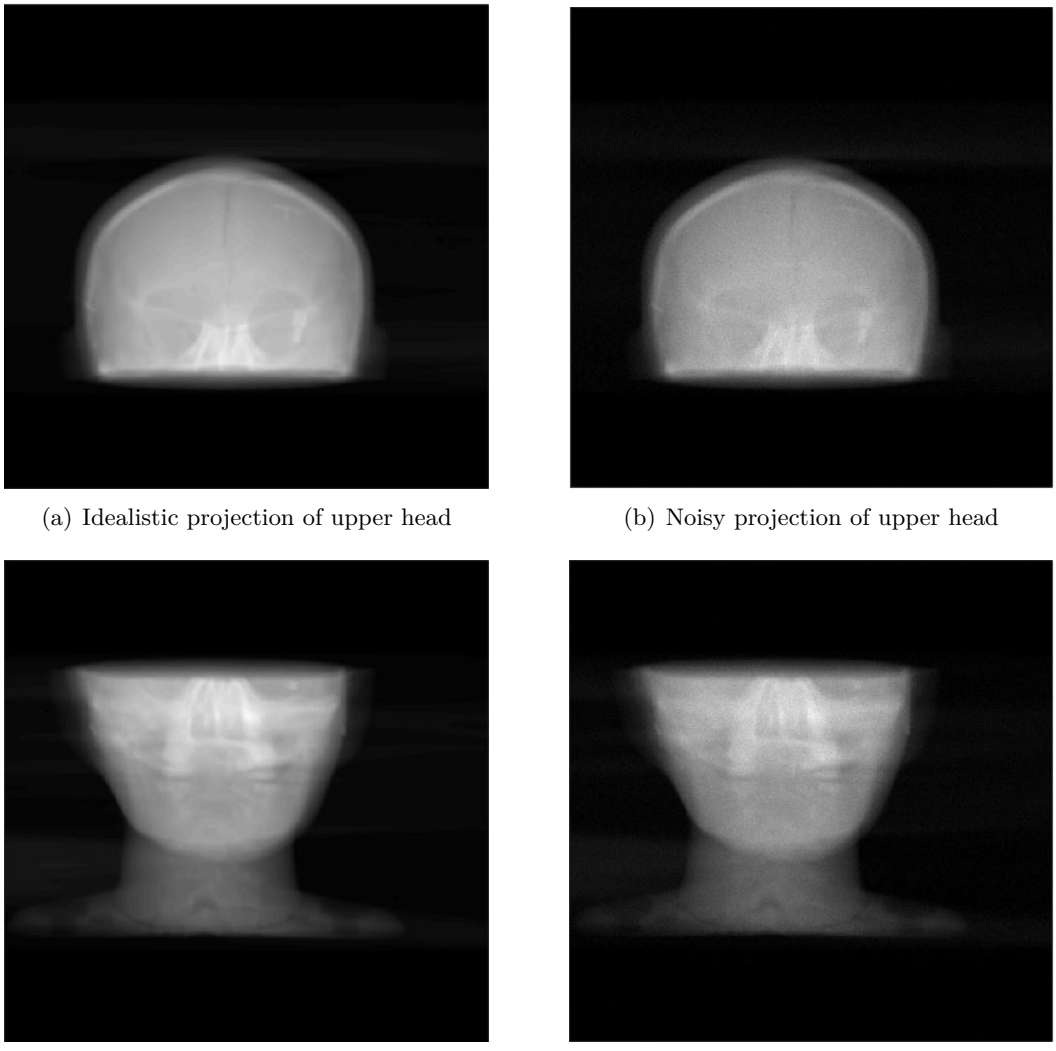
3.2.2 Workflow starting from reconstructed images

The main difference with the previous approach is the starting point: instead of starting from projections we start from reconstructions. The reconstructions are in a DICOM (Digital imaging and communication in medicine) format, an international standard that permits to transmit, store, retrieve, print, process, and display medical imaging information. [17]

The acquisition of the reconstructions of the head has been done in two parts. The upper part of the head and the lower part are separated, as shown on Figures 3.3(a) and 3.3(c). So every step of the workflow needs to be computed separately for both head parts. The acquisition has been performed with a Philips CBCT and reconstructed with an algorithm close to FDK.

The first step consists in recreating the projections. This is done by the function `projectionsdcm` shown in Appendix B.3.1, that transforms the DICOMS into a 3D-matrix of projections. To do so, the function of TIGRE Ax is used. The `Ax` function mimics the CBCT geometry (chosen by the definition of a geometry), but these projections are idealistic: no scattering, no electronic noise in the detector. Moreover it assumes monochromatic x-rays which is not the case in a real machine. Lastly it is not showing photon counts but already linearized attenuation. For more realistic projections Monte Carlo simulations would be needed.

Here, two sets of projections were created: without noise (the idealistic projections) and the case with added noise. The added noise is represented by two components: Gaussian and Poisson noise. The first one is related to the possible electronic noise of the detector. The Poisson component is related to the maximum photon count. Both noisy and not noisy projections are shown on Figures 3.3(c), 3.3(d), 3.3(a) and 3.3(b).



(a) Idealistic projection of upper head

(b) Noisy projection of upper head

(c) Idealistic projection of lower head

(d) Noisy projection of lower head

Figure 3.3: Projections created from DICOMs

After the projections have been created, the reconstruction part can start with the 3D-matrix of projections, the defined geometry and the projection angles. The reconstruction algorithms that

have been used for the reconstruction are: FDK, SART, OS-SART, SIRT, CGLS, ASD-POCS and MLEM. The complete workflow starting from reconstructions is showed in Figure 3.4.

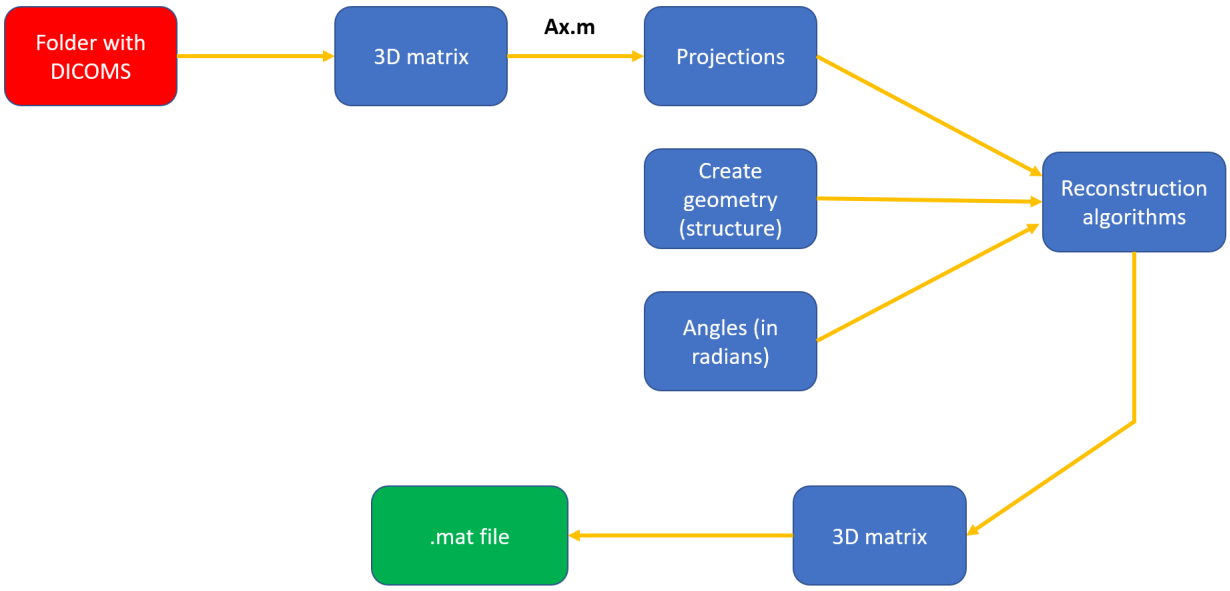


Figure 3.4: Workflow with DICOM reconstructions

3.3 Limitations and advantages of TIGRE

In the sections above the workflow and the TIGRE toolbox have been detailed. The limitations and advantages of the workflow will be made clear trough the results and the discussion of these results in Chapters 4 and 5. This section will try to explain some limitations and advantages of the TIGRE toolbox.

First of all TIGRE is a very complete and easy to use toolbox. It has a lot of algorithms implemented and many examples of scripts and additional tools complementary with the algorithms: image quality metrics functions, denoising functions, 3D-plot functions for projections, etc.

The toolbox is also persistently being improved and completed by the creators. This means that new features are added on a regular basis or that algorithms are being optimised through updates on their GitHub account `https://github.com/CERN/TIGRE`.

Additionally, if you encounter any problem or have any question you can easily contact them and they are very responsive.

In contrast there are also some limitations to the TIGRE toolbox. First, the fact that some functions are run on GPU and therefore need highly performant hardware especially the graphic card which is not available in all standard computers. Next, even though you have the computer with the needed features it still could not be sufficient. If the version of your graphics card is too old for example it will limit your GPU access as it has not the sufficient memory. For example the SART-TV algorithm needs to use 3 mex-files: one for the Ax function, one for the Atb function and one for the TV-denoising function. This is not possible with older graphics cards.

Another possible problem is the amount of parameters that TIGRE can handle. This will be shown more explicitly in Section 4.1 in the case of offsets within the projections.

3.4 Testing

After having produced the 3D-matrices of reconstructions as explained in Section 3.2 the image quality can start to be assessed. This is done both by visual observation and the computation of some metrics. The metrics are tested through a script. The obtained results are detailed in Chapter 4.

Additionally, the influence of some parameters has to be tested. First of all the effect of noisy projections on the reconstruction. Next, how the reconstructions are affected by the number of iterations and lastly the influence of the amount of projections for example. The metrics of all these reconstructions will be discussed in Chapter 4. To objectively asses the image quality, the following four metrics were measured: Structural Similarity (SSIM), Root Mean Square Error (RMSE), Correlation Coefficient (CC) and the Universal Quality Index.

3.4.1 Image quality metrics

Structural Similarity Index [7]

This index is based on human preferences. The human visual system tends to focus on the structural information of an object. So by comparing the structural similarity between two images you can get a good indication of the perceptual image quality. Moreover, SSIM focuses less on the distortions that do not influence the structure of the image.

Mathematically the SSIM index is formulated as follows. x and y being the two images: $\mathbf{x} = \{x_i | i = 1, ..., M\}$ and $\mathbf{y} = \{y_i | i = 1, ..., M\}$. The SSIM is then defined as:

$$S(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{3.1}$$

With C_1 and C_2 small positive constants, μ_x and μ_y the means of $\mathbf{x}$ and $\mathbf{y}$ respectively, σ_x and σ_y the variances of $\mathbf{x}$ and $\mathbf{y}$ respectively and σ_{xy} the covariance of $\mathbf{x}$ and $\mathbf{y}$.

The maximum structural similarity is 1 and is obtained if and only if both images are identical.

Root Mean Square Error

This is a metric that is used in a lot of different contexts. In the case of images it is comparing pixel by pixel the error and then normalising the square of these errors. Hence, if we have two images as defined above then the MSE is defined as:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(x_i - y_i)^2 \tag{3.2}$$

The Root Mean Square Error is the square root of the MSE. The advantage of taking the square root is to convert the values back to the same unit as the images. In the case of the reconstructions that will be produced this means that the RMSE will be in Hounsfield Units (HU), which is more easy to interpret.

HU [1] depends on the attenuation coefficient of the traversed material. The HU of a material is computed as described in equation 3.3. Some examples of HUs for material are: water which is 0 HU, air which correspond to -1000 HU or bone that has a corresponding HU of 200 to 1000.

$$HU = 1000 * \frac{\mu - \mu_{water}}{\mu_{water} - \mu_{air}} \tag{3.3}$$

The minimum RMSE is 0, this means that both pixels (or voxels) have the same HU value. This is the ideal situation as we want the reconstruction to be as close as possible to the reference image (the real value within the body).

To be able to compute the RMSE in HU a transformation is needed. The DICOM format gives some raw data between 0 and 65534, which is obviously not in HU. So these values simply need to be rescaled following the following formula $Ax + B$ with x the DICOM image, A a multiplying factor called the *slope* and B a factor that is called the *intersect* and here values -1000. This transformation need to be done on both the DICOMs as well as on the reconstructed images produced by the algorithms. So when computing the RMSE it is in HU, which can be physically interpreted.

Correlation Coefficient [20]

The Pearson's linear Correlation Coefficient is a tool to calculate the correlation between two images. The correlation measures the linear dependency between two variables, and in the contexts of medical imaging it shows how linear dependant both images are on each other.

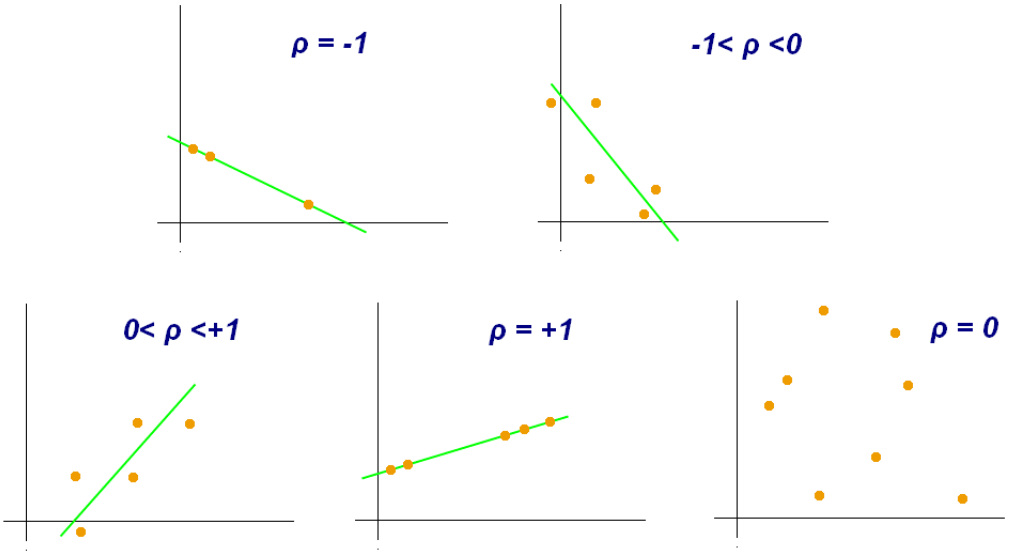


Figure 3.5: Different values of Correlation Coefficients [27]

Practically the Correlation Coefficient is defined as follows:

$$\rho = \frac{COV(X,Y)}{\sigma_x \sigma_y} \tag{3.4}$$

With $COV(X,Y) = E[(X - E[X])(Y - E[Y])]$.

The value of the Correlation Coefficient varies between -1 and 1. The different possibilities are shown on Figure 3.5. When ρ equals 0 we see that there is no linear dependence and in the case of two images this means that pixels do not have any link which would imply that one or both images would contain significant errors.

Universal Quality Index [5]

The Universal Image Quality Index (UQI) is a bit different from the classical error summation methods such as MSE or RMSE. Figures 3.6(b), 3.6(c) and 3.6(d) below show how images which all have the same MSE are not perceived with the same quality. All these three images have an MSE of 225. But when looking at them you see a clear difference in quality. Enhancing the contrast has the same impact on the MSE of a picture as blurring but the picture actually looks a lot sharper and the shapes, such as the eyes, are far more distinguishable. When adding Gaussian noise a lot of small dots appear on the figure, but again here the impact on the MSE is the same.

And this is the critic that the authors of the article [5] had on the MSE metric. So they tried to implement another metric that modelised the image distortation as a combination of three factors: loss of correlation, luminance distortion and contrast distortion. Like MSE it is mathematically based but it fits better with the human perception. And this can be illustrated by Figures 3.6(b), 3.6(c) and 3.6(d) as there respective UQI are 0.3461, 0.3891 and 0.9372. This corresponds to the human perception.

(a) Initial image

(b) Blurred image

(c) Image with Gaussian Noise

(d) Contrast stretched image

Figure 3.6: Comparing initial and distorted images. [5]

Mathematically the UQI is formulated as follows, with x the initial image and y the tested image:

$$Q = \frac{4\sigma_{xy}\bar{x}\bar{y}}{(\sigma_x^2 + \sigma_y^2)(\bar{x}^2 + \bar{y}^2)} \tag{3.5}$$

With $\bar{x}$ and $\bar{y}$ being the means of x and y, σ_x^2 and σ_y^2 the variances of x and y and σ_{xy} the covariance of x and y.

The range of Q goes from -1 to 1.

Rewriting the equation 3.5 as follows:

$$Q = \frac{\sigma_{xy}}{\sigma_x\sigma_y} \cdot \frac{2\bar{x}\bar{y}}{\bar{x}^2 + \bar{y}^2} \cdot \frac{2\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2} \tag{3.6}$$

We see the three parts: the first is the correlation coefficient (discussed in Section 3.4.1), the second fraction is comparing the mean luminances (the range is between 0 and 1) and the last fraction is comparing the similarity of the contrast (the range is also between 0 and 1).

It is interesting to notice that UQI is similar to SSIM as C1 and C2, the two constants in the SSIM equation 3.1 are small constants. If both constants were equalt to zero, the metric would be identical.

Other image quality metrics

The four presented metrics are mathematically based. Though UQI and SSIM try to put the stress on perception they still are mathematically based. Their advantage is that they are quite easy to compute. Additionally they are objective, not depending on the individual observers. [5] Other objective image quality metrics also exist, for example PSNR (Peak Signal to Noise Ratio).

But with images the perception of the people using the images is still crucial and that is why human observation is also an important metric. Section 3.4.1 showed that, for example, the same MSE does not mean the same (percepted) image quality. This is why some subjective metrics try to base themselves on a human visual system model. [5]

3.4.2 Other metrics

Besides the image quality, other critical metrics are needed to be taken into account: time and memory consumption for example. As mentioned in the introduction, time is crucial as CBCTs are mostly used in per-treatment phase, which means that the reconstruction can not be too time consuming.

Moreover for the iterative algorithms there is the time that each iteration needs, which will be crucial in the trade-off between time and image improvement through an additional iteration.

Lastly, the context where the images are used in is also crucial. Some applications need really high precision reconstruction, while other just a fast reconstruction. This is also to be taken into account.

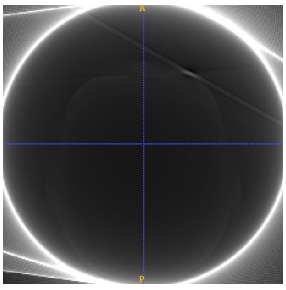
Chapter 4

Results

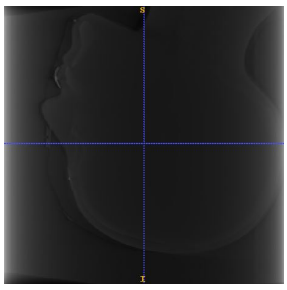
The following sections will show the results produced by the different reconstruction algorithms. Different parameters have been tested: number of iterations (in the case of iterative algorithms) and number of projections. As explained in Chapter 3 two data sets were used and the results of both will be described here. The results will be discussed in Chapter 5.

4.1 Results starting with projections

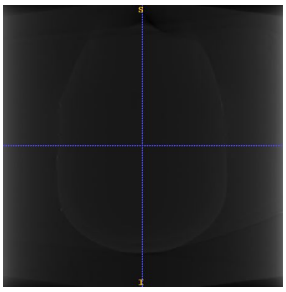
The results produced using the HIS projections were not the ones expected. The number of projections is 196, but not perfectly distributed as the angles vary from $-179,9215°$ and $24.8623°$. In Figures 4.1, 4.2 and 4.3, we can barely distinguish the head and it does not give any anatomical details. Such poor reconstructions are not useful in the context of medical imaging.



(a) Transverse plane



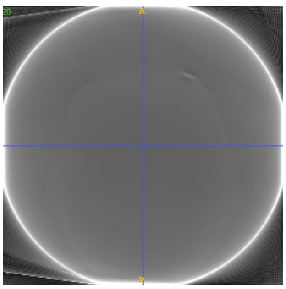
(b) Sagittal plane



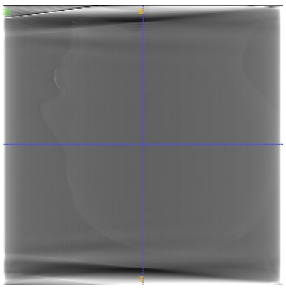
(c) Coronal plane

Figure 4.1: Reconstruction with the FDK algorithm from HIS file projections

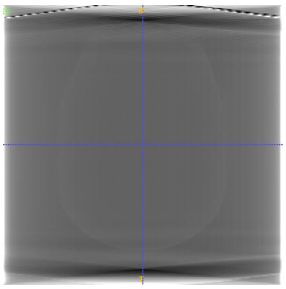
Figure 4.1 shows the reconstruction with the classic FDK algorithm as Figure 4.2 shows the results of the reconstruction with the OS-SART algorithm. Even if we increase the number of iterations we cannot improve the quality of the reconstruction (this is showed on Figure A.1 in Appendix A).



(a) Transverse plane



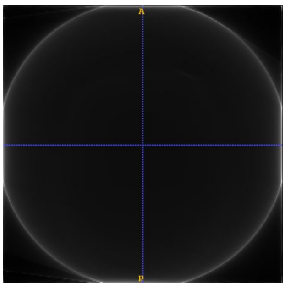
(b) Sagittal plane



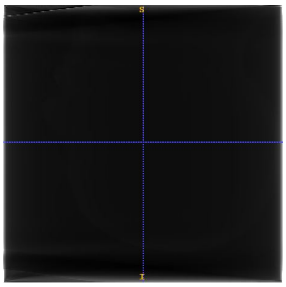
(c) Coronal plane

Figure 4.2: Reconstruction with the OS-SART algorithm (20 iterations) from HIS file projections

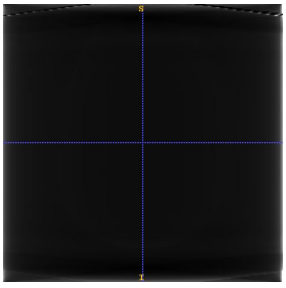
Figure 4.3 (MLEM reconstruction with 40 iterations) is a last example of the reconstruction starting with projections from the HIS files. This confirms the bad handling of the offsets as there seems to be no visible reconstruction at all.



(a) Transverse plane



(b) Sagittal plane



(c) Coronal plane

Figure 4.3: Reconstruction with the MLEM algorithm (40 iterations) from HIS file projections

The reason for such poor results has been mentioned in Section 3.3. As TIGRE is a quiet complete and easy to use toolbox, it cannot handle all possible parameters. The projections contained within the HIS files have each a different offset. TIGRE could handle a global offset for all the projections as the offset is a parameter of the geometry (shown in B.1 in Appendix

B). But it cannot handle an individual offset for each projection. This results in a poor image quality, or even no image at all in the case of MLEM reconstruction.

4.2 Results starting with reconstructions

The projections that were obtained starting from the DICOM reconstructions have been shown in Figure 3.3. Afterwards they have been reconstructed by different algorithms of the TIGRE toolbox.

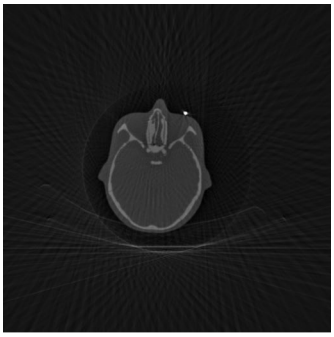
4.2.1 Varying the number of iterations

In this section the reconstructions have been produced with 100 projections, uniformly distributed between 0° and 360°. The reconstructions were tested with 5, 10, 20, 30 and 40 iterations, except for FDK the only non-iterative reconstruction algorithm discussed here.

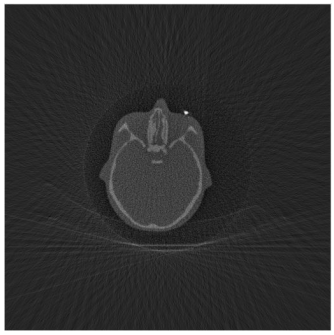
As the upper part of the head has identical reconstruction properties, the next sections will focus on the lower part of the head. We expect to obtain similar results with the upper part of the head. The reconstructions from the ideal and the noisy projections will both be shown by the different number of iterations.

Figure 4.4 shows the FDK reconstruction of the lower part of the head with ideal projections (Figure 4.4(a)) and noisy projections (Figure 4.4(b)). Figure 4.4(c) shows a reconstruction for the upper part of the head with ideal projections (Figure 4.4(c)). FDK is reconstructing very well with ideal projections, but Figure 4.4(b) shows that it struggles to erase the noise of the reconstructions with the noisy projections. Even if it has an influence on the quality, the reconstruction looks sharp and satisfactorily detailed.

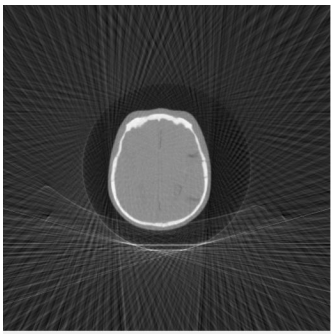
Some artifacts are visible as the bright white point on Figures 4.4(a) and 4.4(b) on the front side of the head, and will be visible on all reconstructions, or the lines that appear in all these reconstructions.



(a) From ideal projections



(b) From noisy projections



(c) From ideal projections

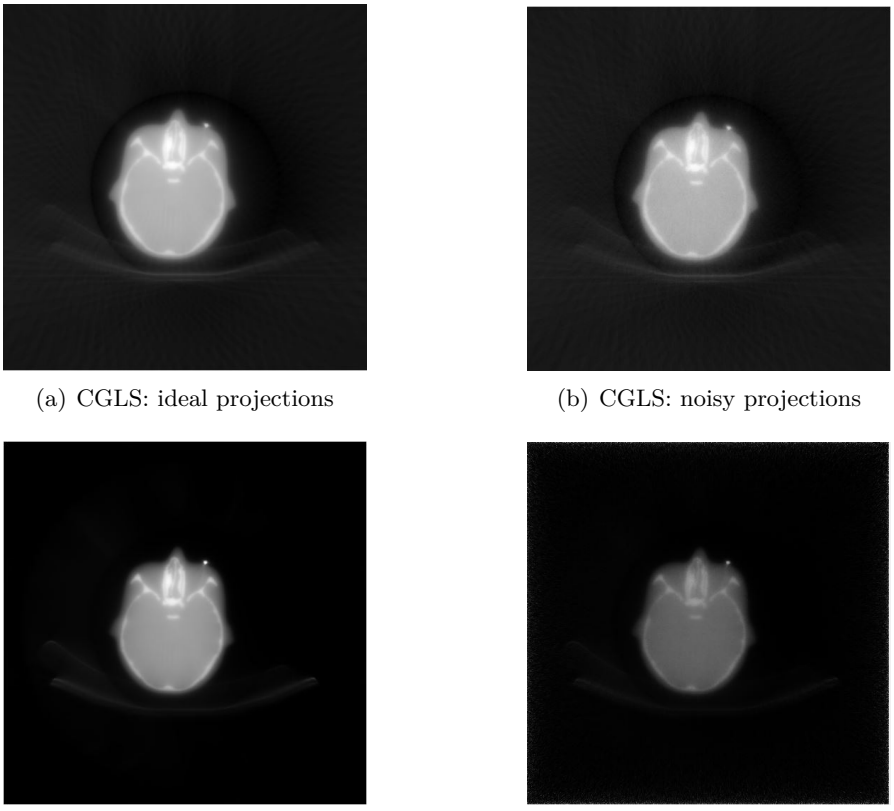
Figure 4.4: Reconstructions with FDK

Reconstructions with 5 iterations

The four images represented in Figure 4.5 are obtained with the CGLS and MLEM algorithms and show similar quality for both cases: reconstructions from noisy and ideal projections. The difference lies within the brightness as the MLEM reconstructed images are darker. But if we compare them with reconstructions from another algorithm, SART for example, we can see that there is still a lot of improvement possible.

On all images some lines are present below the head. This is the support the head phantom was placed on, but not all algorithms show it equally. The MLEM reconstruction technique seems to be able to fade the lines away, while in the images produced by CGLS they are strongly visible.

Looking at the reconstructions created from the noisy projections, both algorithms do not seem to be affected by noise. But this could be due to the fact that MLEM and CGLS would need more iterations to get the same spatial resolution as other algorithms, like SART for example described further on. The spatial resolution refers to the sharpness of an image, meaning the amount of details that can be differentiated within that image. Figures 4.5(a), 4.5(b), 4.5(c) and 4.5(d) are blurry which indicates a lower spatial resolution. However MLEM reconstructions look a little bit sharper as the withe dot is the brightest on Figure 4.5(c).



(a) CGLS: ideal projections

(b) CGLS: noisy projections

(c) MLEM: ideal projections

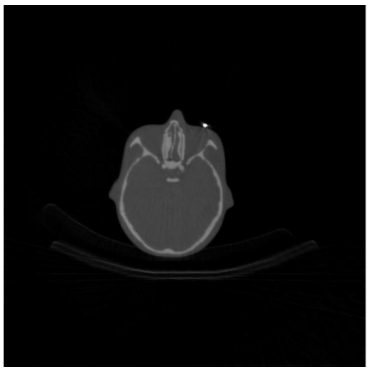
(d) MLEM: noisy projections

Figure 4.5: Reconstruction with CGLS and MLEM algorithm: 5 iterations

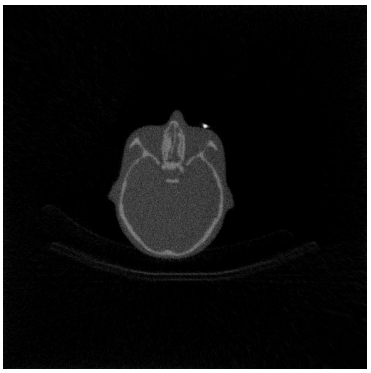
Figure 4.6 shows the reconstructions of the lower part of the head by the SART, OS-SART and SIRT algorithms. The SART reconstruction shows already a good image quality with only 5 iterations. While the OS-SART reconstructed images shown on Figures 4.6(c) and 4.6(d) are of similar quality as the reconstructions from the CGLS algorithm: bright but not sharp. The artifact showing the lines is more noticeable for OS-SART than CGLS. The SIRT reconstructions (Figures4.6(e) and 4.6(f)) are even brighter and blurrier and do not give a clear view of the details in the center of the head.

As we look at Figure 4.6(b), the noise (salt and pepper) is visible within the head. This is due to the sharpness of the image which is significantly higher than for other algorithms. Consequently, this does not only increase the level of detail for the wanted parts (the bones etc.) but also the noise.

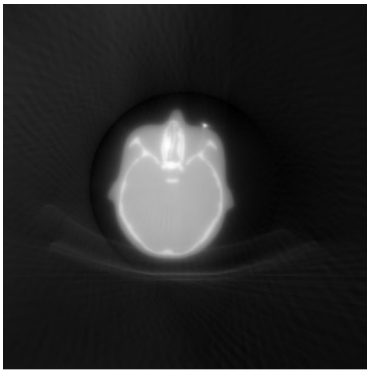
The lines of the support are the most visible in the OS-SART images. The SART reconstructions seem not to get completely rid of them: the lines are still visible but less in comparison with the OS-SART and the SIRT reconstructions.



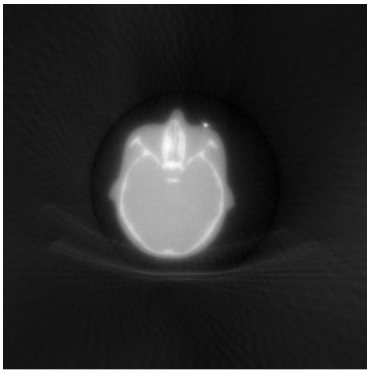
(a) SART: ideal projections



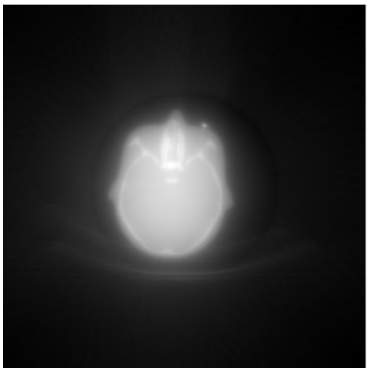
(b) SART: noisy projections



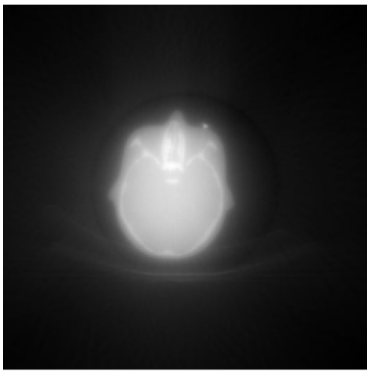
(c) OS-SART: ideal projections



(d) OS-SART noisy projections



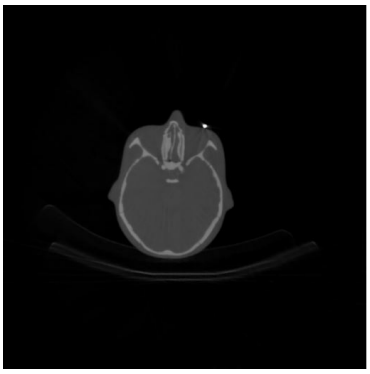
(e) SIRT: ideal projections



(f) SIRT: noisy projections

Figure 4.6: Reconstruction with SART, OS-SART and SIRT algorithm: 5 iterations

ASD-POCS reconstructions (Figure 4.7) have similar results to the SART algorithm. The spatial resolution is good as the details can be distinguished. The sharpness is confirmed by the reconstructions from the noisy projections as the salt and pepper dots are visible within the head. Moreover, we see that the support is slighlty visible, as it was for SART which enhances the fact that the SART and ASD-POCS produce similar reconstructions.



(a) ASD-POCS: ideal projections

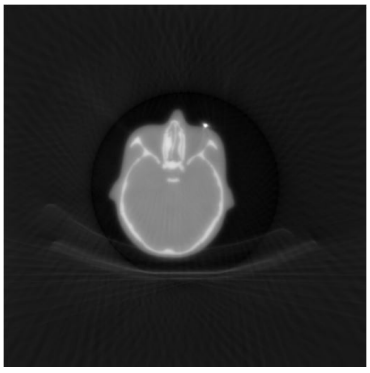
(b) ASD-POCS noisy projections

Figure 4.7: Reconstruction with the ASD-POCS algorithm: 5 iterations

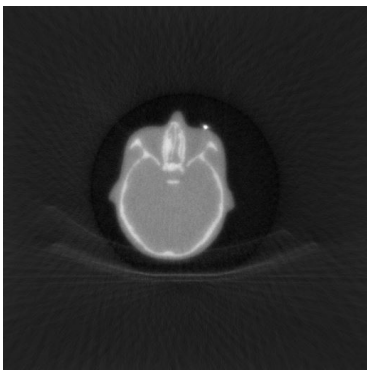
Reconstructions with 10 iterations

Doubling the number of iterations should improve the image quality of most of the reconstructions. The two algorithms that already gave satisfaction with 5 reconstructions were SART and ASD-POCS. This is confirmed by Figures A.2(a), A.2(b), A.2(c) and A.2(d) in Appendix A, produced with 10 iterations, that are almost identical to the ones produced with 5 iterations. Both algorithms converged very quickly, based on the number of iterations.

The improvement due to more iterations is visible for the CGLS algorithm. As the reconstructions with 5 iterations, displayed on Figure 4.5 are blurry the reconstructions of Figure 4.8 are looking sharper: better contrast and clearer borders. If we look at the details near the noise, the CGLS images with 10 iterations look more precise. This increased sharpness makes also the lines of the support more visible within the CGLS reconstructions. Moreover it enhances the presence of the noise, as is demonstrated on Figure 4.8(b).



(a) CGLS: ideal projections



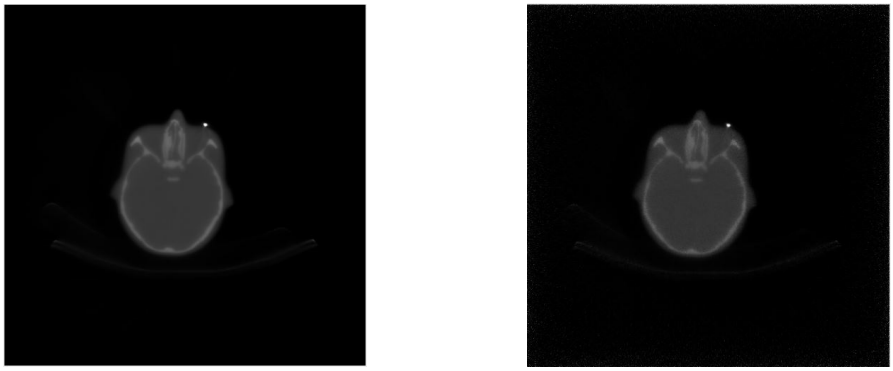
(b) CGLS: noisy projections

Figure 4.8: Reconstruction with the CGLS algorithm: 10 iterations

The reconstructions produced by the MLEM algorithm also improves with the increased number of iterations. They do not improve as much as the reconstructions from CGLS. Looking at the details of the noise it still does not look as precise as the details visible on Figure 4.7, the ASD-POCS recontruction with 5 iterations.

In contrast with the CGLS algorithm, the MLEM algorithm got rid of the lines below the head on Figures 4.9(a) and 4.9(b). This is remarkable as it is a physical object. Moreover we can see that the noise pattern, which is visible in most of the other algorithms within the head, is not visible within the MLEM reconstructions. This could be due to lack of sharpness or the

algorithm could handle the noise very well. By increasing the number of iterations this will become clearer.

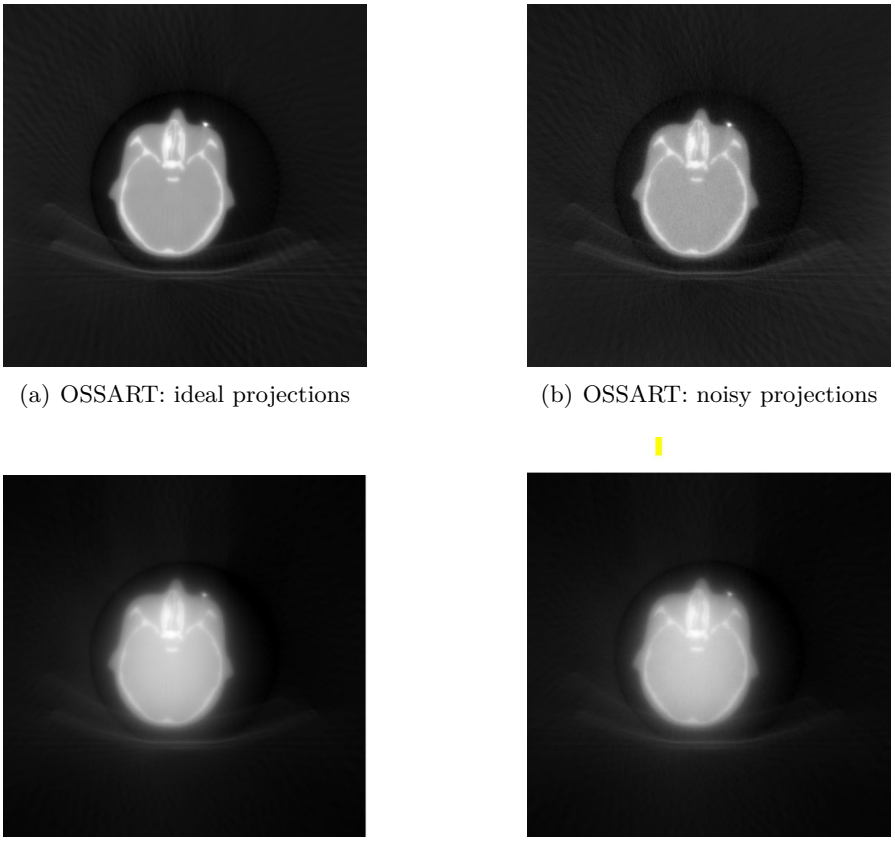


(a) MLEM: ideal projections (b) MLEM: noisy projections

Figure 4.9: Reconstruction with the MLEM algorithm: 10 iterations

The reconstructions obtained with OS-SART (Figures 4.10(a) and 4.10(b)) are similar to the reconstructions from the CGLS algorithm in Figure 4.8. Due to the better sharpness, the noise got also intensified. The lines crossing the images are the most visible on these two reconstructions.

Lastly, the SIRT reconstructions in Figures 4.10(c) and 4.10(d) show little improvement. The spatial resolution is still too low for the images to be useful within a medical context. The noise and the lines are less visible than in the OS-SART images, but this is probably due to the lack of sharpness rather than to the algorithm.



(a) OSSART: ideal projections (b) OSSART: noisy projections

(c) SIRT: ideal projections (d) SIRT: noisy projections

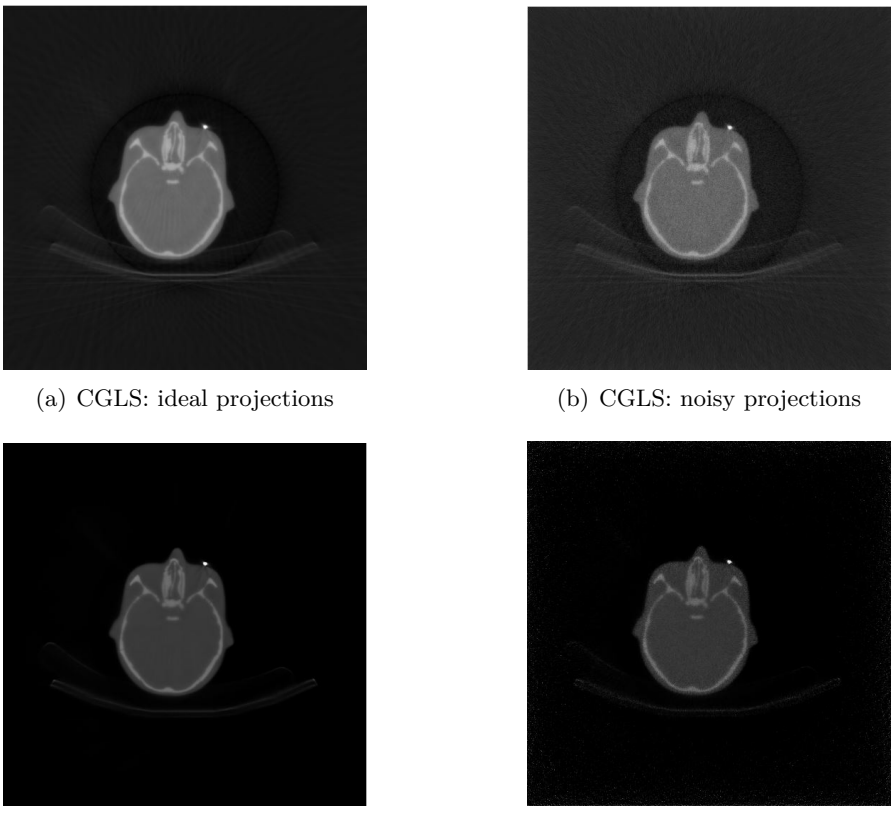
Figure 4.10: Reconstruction with OS-SART and SIRT algorithm: 10 iterations

Reconstructions with 20 iterations

Reaching 20 iterations, the reconstructions are expected to become accurate not needing much more improvement. ASD-POCS and SART have already proven to be performing with 5 and 10 iterations.

In the case of CGLS, the reconstructions with 17 iterations are similar to the ones with 10 iterations. The images have become a bit sharper, which is not visible for the reconstructions from ideal projections. The noise is more visible on Figure 4.11(b) with respect to the reconstruction with 10 iterations. The CGLS reconstruction starting from the noisy projections (Figure 4.11(b)) has not been possible with 20 iterations as the algorithm stopped at 17 iterations due to divergence. This means that the reconstructions stopped improving and the algorithm, therefore, stopped too. Within the algorithm this is implemented through a stop criterion present in the loop conditions (in addition to the maximal number of iterations). It stops the algorithm in case the error increases.

The MLEM reconstructions, however, show some improvement as Figure 4.11(c) is a lot sharper and the details of the bones are clearly visible. Moreover the salt and pepper pattern of the noise is not as visible as in the other reconstructions. This confirms that MLEM looks to handle noise really well.



(a) CGLS: ideal projections

(b) CGLS: noisy projections

(c) MLEM: ideal projections

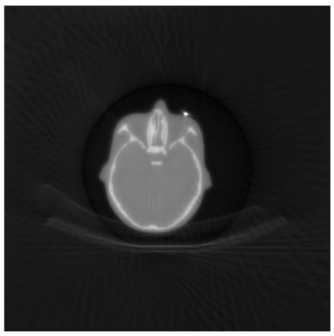
(d) MLEM: noisy projections

Figure 4.11: Reconstruction with CGLS and MLEM algorithm

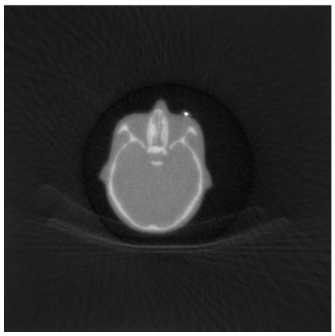
The OS-SART reconstructions have improved. While for 5 and 10 iterations they had a similar quality to the CGLS reconstructions, with 20 iterations they seem to have outperformed the CGLS reconstructions. The colors are brighter, the images look sharper and the noise is less visible.

Figures 4.12(c) and 4.12(d) show that, for the SIRT reconstructions, the blurriness has diminished and the edges start to stand out. This is an improvement but still SIRT is the worst

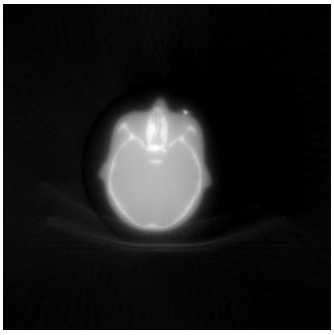
reconstruction and seems not to be able to get a sufficient high spatial resolution for a medical use.



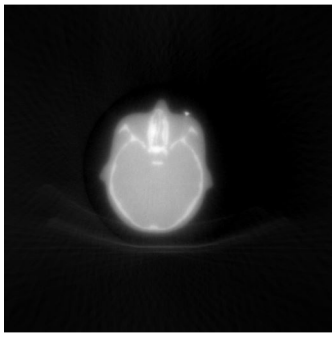
(a) OS-SART: ideal projections



(b) OS-SART: noisy projections



(c) SIRT: ideal projections

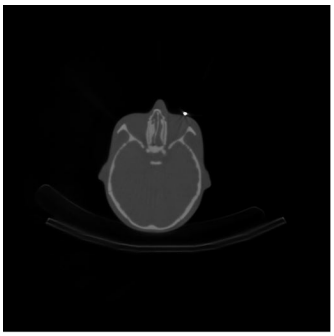


(d) SIRT: noisy projections

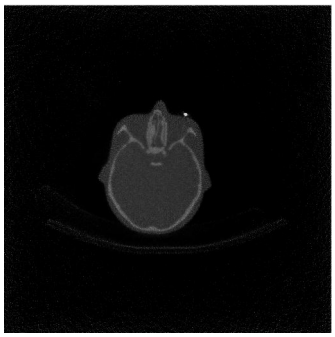
Figure 4.12: Reconstruction with OSSART and SIRT algorithm: 20 iterations

Reconstruction with 40 iterations

Figure 4.13 shows that MLEM has converged as it has little difference with the reconstructions with 20 iterations (Figures 4.11(c) and 4.11(d)). The only visible difference is the sharpness that has slightly improved. But this enhances (slightly) the noise too. Although, MLEM looks like the best algorithm regarding noise supression. The increased sharpness is also visible with the support being more visible below the head.



(a) MLEM: ideal projections

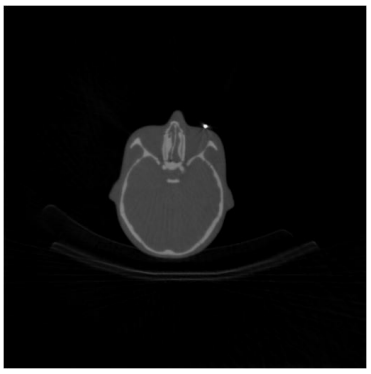


(b) MLEM: noisy projections

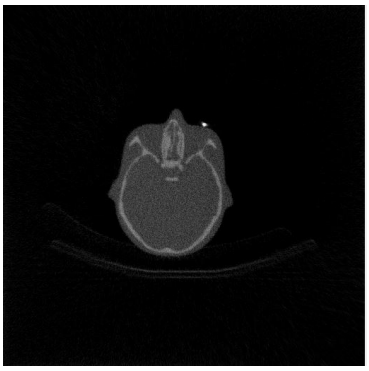
Figure 4.13: Reconstruction with MLEM algorithm: 40 iterations

Figure 4.14 shows the reconstructions with 40 iterations for OS-SART and SIRT. OS-SART (Figures 4.14(a) and 4.14(b)) shows similar results to the ones obtained with SART with 5

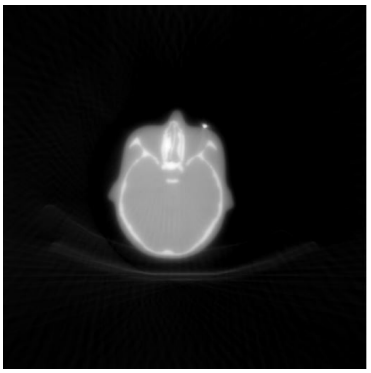
iterations. The reconstructions from ideal projections look flawless but the fact that it has such a great spatial resolution makes the noise more visible in the case of noisy projections. The SIRT (Figures 4.14(c) and 4.14(d)), the less convincing algorithm, gave better results. But the reconstruction remains blurry even with this significant increase in iterations.



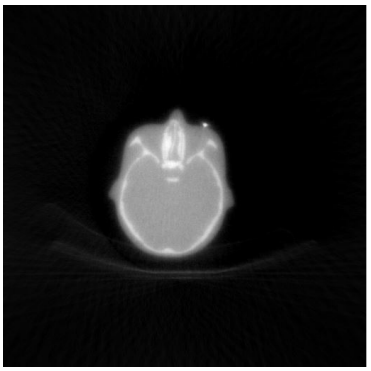
(a) OS-SART: ideal projections



(b) OS-SART: noisy projections



(c) SIRT: ideal projections



(d) SIRT: noisy projections

Figure 4.14: Reconstruction with OS-SART and SIRT algorithm: 40 iterations

4.2.2 Varying number of projection angles

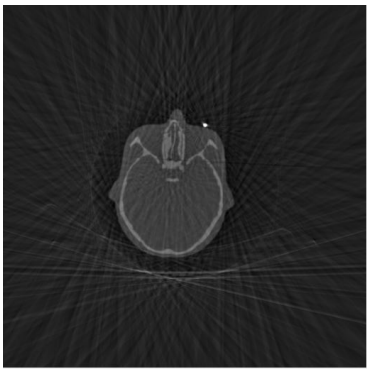
Iterative algorithms are known to perform better with noise and artifacts in comparison with FDK. The main interest of decreasing the amount of projections in comparison with the amount needed for FDK to produce good results is to reduce the amount of radiation send to the patient.

This section will look at the influence of the number of projections on the reconstructions. The different number of projections that have been tested are 50, 100 (which have been presented in Section 4.2.1) and 200.

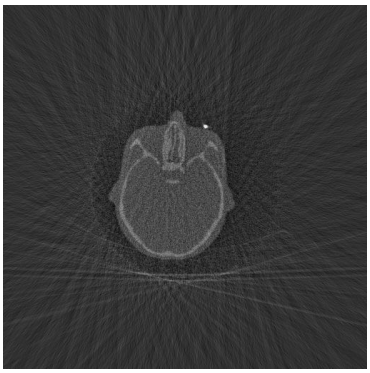
FDK

Figure 4.4(b) showed the great influence of noise on the quality of the reconstruction when it was generated by 100 projections. Figure 4.15 shows that increasing the number of projections has a significant influence on the image quality of the ideal and noisy case.

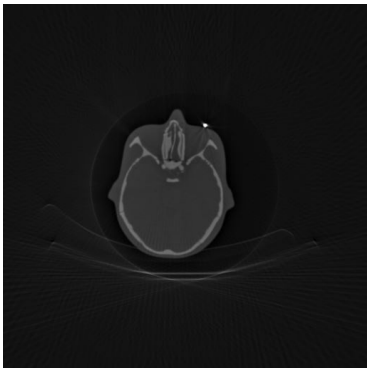
Figure 4.15(d) shows little noise within the reconstruction while keeping sharp and detailed. If we increase the number of iterations to 360 (Figures 4.15(e) and 4.15(f)) we see no visible further improvement. While lowering the number of projections to 50 has a negative influence on the quality, as the images are filled with lines. The withe dot is also visible here.



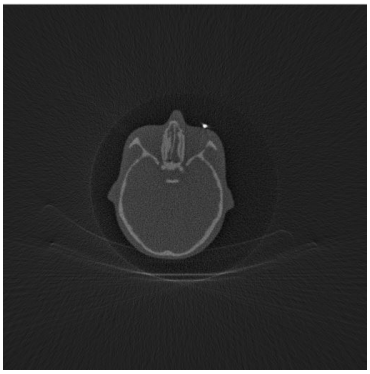
(a) FDK: 50 ideal projections



(b) FDK: 50 noisy projections

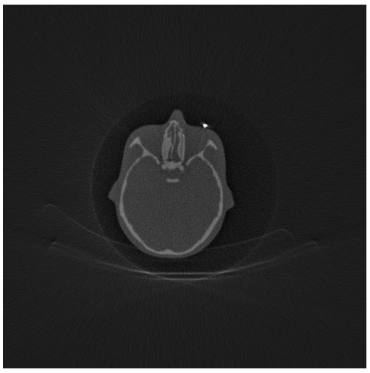


(c) FDK: 200 ideal projections



(d) FDK: 200 noisy projections

(e) FDK: 360 ideal projections



(f) FDK: 360 noisy projections

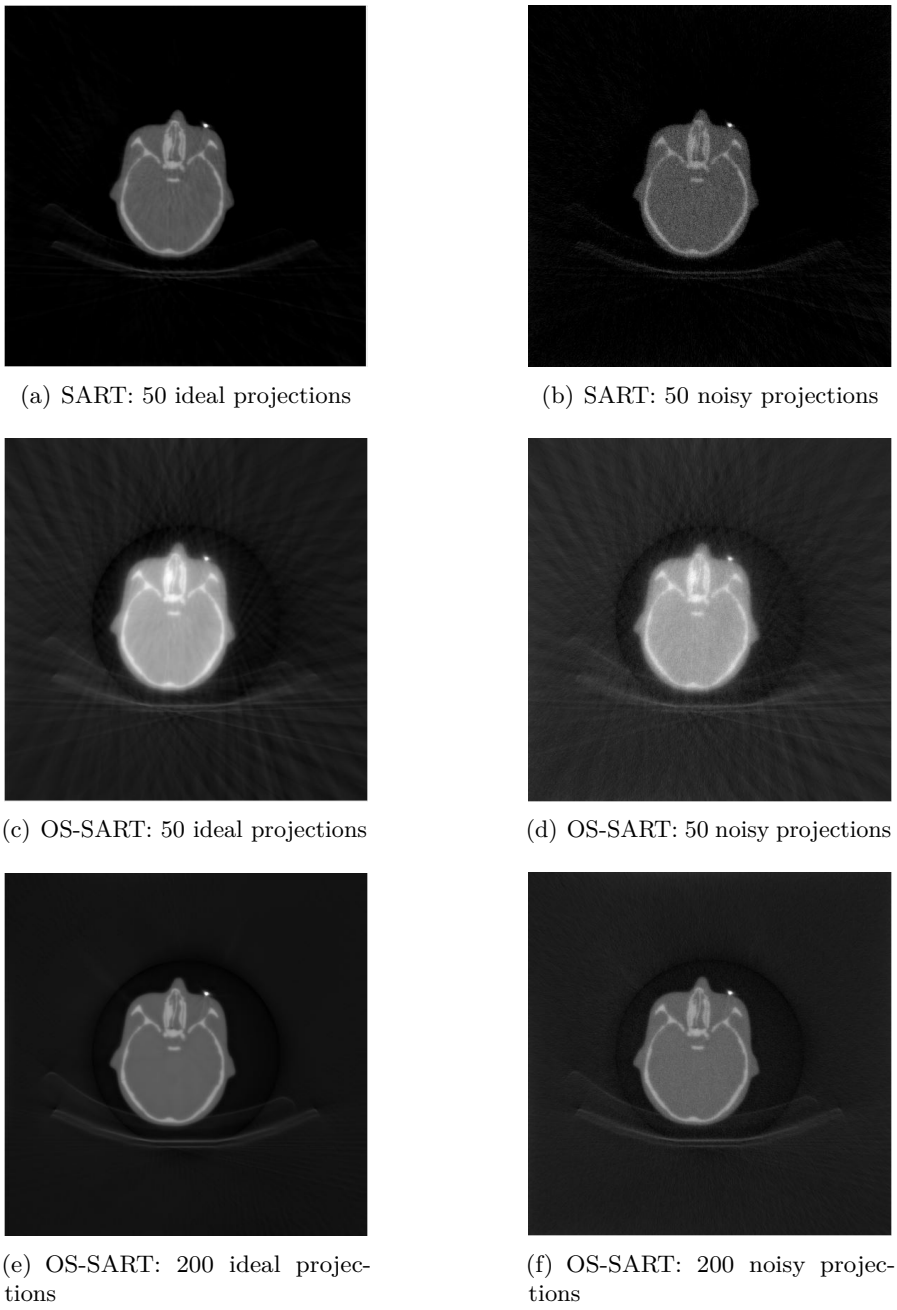
Figure 4.15: Reconstruction with FDK for different number of projection angles

SART, OS-SART and SIRT

SART has already good results with 5 iterations and 100 projections (see above in Section 4.2.1). Increasing the number of projections is therefore not interesting. Figures 4.16(a) and 4.16(b) show the results of the reconstruction with 5 iterations and 50 projections. Cutting the number of projections by half has no influence on the image quality. The spatial resolution is considerable. Moreover in the case of noisy projections the noise can be seen, due to the sharpness of the image.

The reconstructions with the OS-SART algorithm (Figures 4.16(c), 4.16(d), 4.16(e) and 4.16(f)) were produced with 20 iterations. Decreasing the amount of projections, decreases the image quality. Lines appear through the image, like in the FDK reconstructions (Figures 4.15(a) and 4.15(b)), but here the support can be discerned. Additionally, the noise is more visible in the case of noisy projections and the sharpness is not as good as with 100 projections.

On the other hand increasing the number of projections increases the image quality as Figure 4.16(e) looks better than the SART reconstruction. Figure 4.16(f) shows no noise within the head and a good spatial resolution.



(a) SART: 50 ideal projections

(b) SART: 50 noisy projections

(c) OS-SART: 50 ideal projections

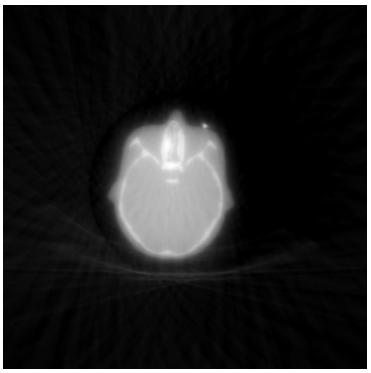
(d) OS-SART: 50 noisy projections

(e) OS-SART: 200 ideal projections

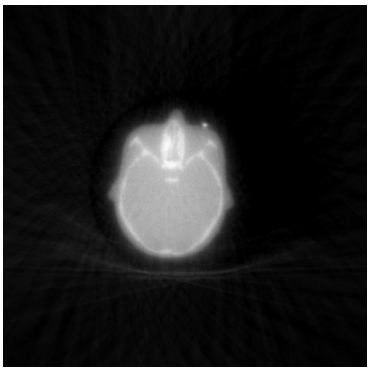
(f) OS-SART: 200 noisy projections

Figure 4.16: Reconstruction with SART and OS-SART for different number of projections

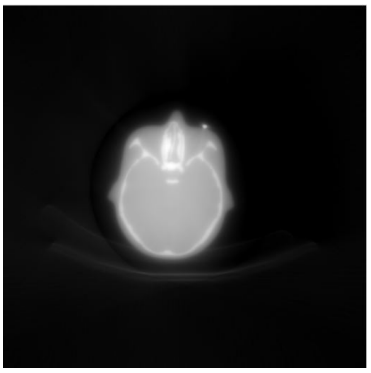
SIRT reconstructions were the least good reconstructions when testing the influence of iterations. Here the reconstructions (Figure 4.17) are created from 20 iterations with 50 and 200 projections. Comparing both we can see that the difference between 50 and 200 projections is not as large as the difference between 10 iterations and 40 iterations showed in Section 4.2.1. Moreover the quality of the image is similar to the image produced with 100 projections. Increasing the number of projections seems to have less influence than the number of iterations.



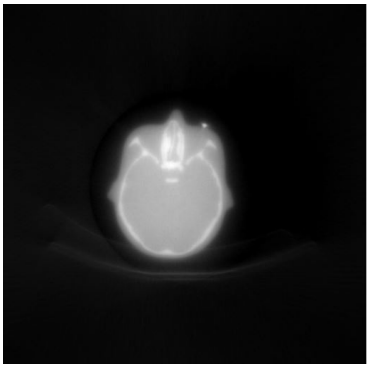
(a) SIRT: 50 ideal projections



(b) SIRT: 50 noisy projections



(c) SIRT: 200 ideal projections



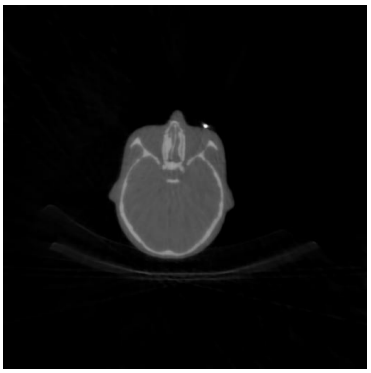
(d) SIRT: 200 noisy projections

Figure 4.17: Reconstruction with SIRT for different number of projection angles

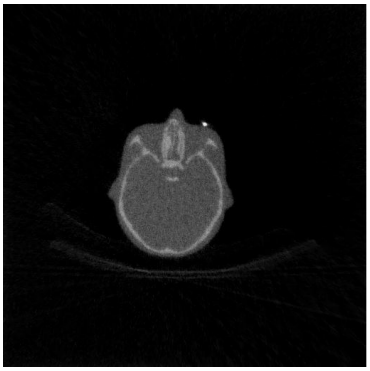
ASD-POCS

ASD-POCS had the similar great results as SART when testing the number of iterations. Figure 4.18 shows the reconstructions with 5 iterations and 50 projections.

Again, the quality is close to the one obtained with SART and even though the number of projections has been cut in half in comparison with Figures 4.7(a) and 4.7(b), the image is still a worthy reconstruction. Even with the 50 noisy projections, the reconstruction is comparable with 100 projections. As for SART (Figure 4.16(b)) noise is visible in Figure 4.18(b).



(a) ASD-POCS: 50 ideal projections

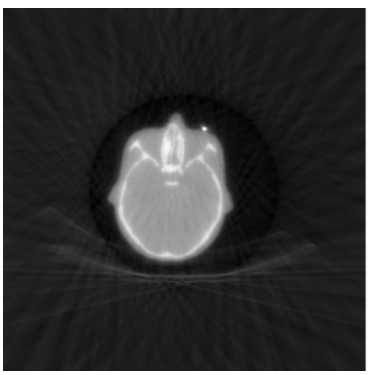


(b) ASD-POCS: 50 noisy projections

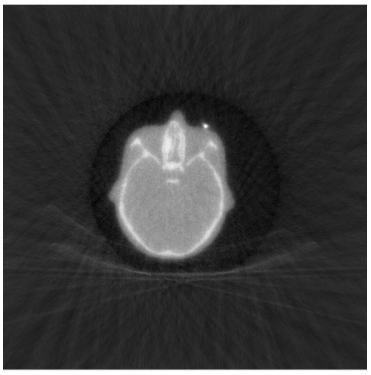
Figure 4.18: Reconstruction with ASD-POCS for different number of projection angles

CGLS

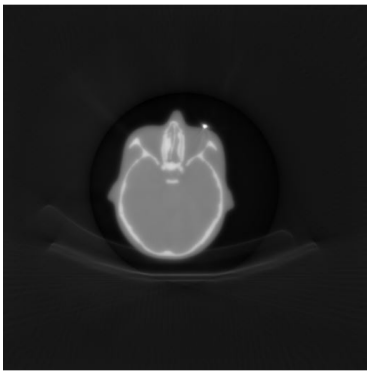
The reconstructions of Figure 4.19 are obtained with 10 iterations and 50 or 200 projections. The results are similar with the ones obtained with the OS-SART algorithm. The number of projections has a significant influence on the quality of the reconstructions. Figure 4.19(d) shows no visible traces of noise while Figure 4.19(b) shows clear noise patterns inside the head. Additionally it makes more lines appear crossing the image.



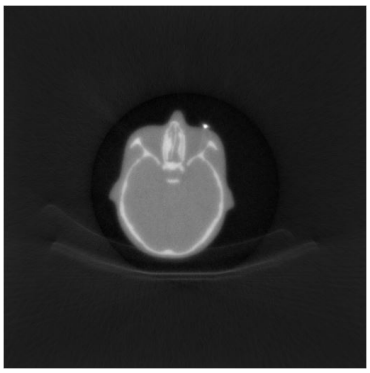
(a) CGLS: 50 ideal projections



(b) CGLS: 50 noisy projections



(c) CGLS: 200 ideal projections



(d) CGLS: 200 noisy projections

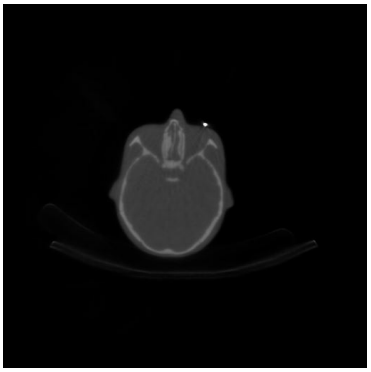
Figure 4.19: Reconstruction with CGLS for different number of projection angles

MLEM

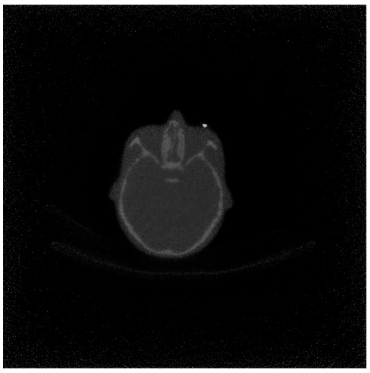
Figure 4.20 shows reconstructions with 20 iterations for 50 and 200 projections in both noisy and ideal cases.

MLEM reconstructions seem not to suffer too much from lowering the number of projections. But there is a loss of sharpness: small details are less visible with less projections. The influence of noise is rather small for both reconstructions (Figures 4.20(b) 4.20(d)).

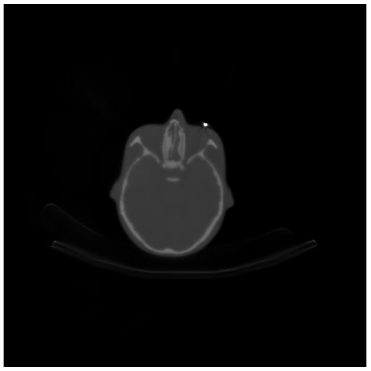
The reconstructions of Figure 4.20 do not show the support.



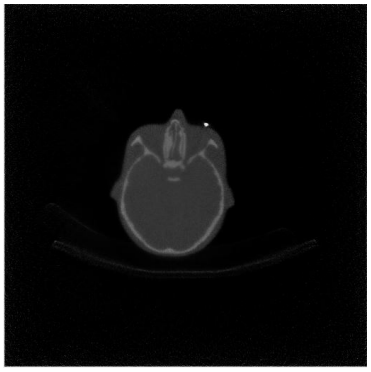
(a) MLEM: 50 ideal projections



(b) MLEM: 50 noisy projections



(c) MLEM: 200 ideal projections



(d) MLEM: 200 noisy projections

Figure 4.20: Reconstruction with MLEM for different number of projection angles

4.2.3 Measured metrics of the reconstructions

The results have been described in Sections 4.2.1 and 4.2.2. In this section they will be compared with the metrics explained in Section 3.4.1. Both the metrics for the variation of the number of iterations and the number of projections will be discussed. All metrics are included in the tables of Appendix A.2.2

The reference image that will be used in the comparisons is the one composed by the original DICOM reconstructions, transformed in HU. One slice (the same as the one chosen in Chapter 4) of the reconstruction is shown on Figure 4.21.

We can see that this reconstruction looks very sharp: clear edges of the bones and details within the head. The good spatial resolution is confirmed by the support of the phantom that looks a lot sharper than in the reconstructions shown in Sections 4.2.1 and 4.2.2.

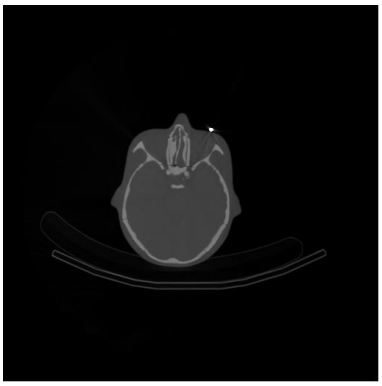


Figure 4.21: Slice of the reference reconstruction

SSIM

This metric confirms the impression given by the images shown in the previous sections: using SART and ASD-POCS algorithm results in better reconstructions. While other algorithms need 10 or more iterations to improve their SSIM, Figure 4.22 shows that SART and ASD-POCS only need 5 iterations to have a really high Structural Similarity (as the ideal value is 1). The SSIM of SART and ASD-POCS do not evolve with the number of iterations, when the reconstructions are performed from ideal projections.

The performance of SIRT also confirms what has been visually assessed in Sections 4.2.1 and 4.2.2. They were visually the worst reconstruction and do not perform either when it comes to the structure of the image. The SSIM of the SIRT reconstructions are improving significantly with the number of iterations in the beginning but starts to flatten after 20 iterations. This is also visible on the reconstructions as the difference between the images of the reconstructions from 5 iterations and 20 iterations is significant, while the difference between 20 and 40 iterations is less important.

In comparison OS-SART, which belongs to the same family of algorithms, has a high and constant SSIM but increases at the end. Theoretically, as mentioned in Section 2.2.1, the OS-SART algorithm should give a better reconstruction in comparison with the SIRT algorithm and this is confirmed based on the SSIM. CGLS had similar reconstructions to OS-SART, this is confirmed by the SSIM.

The FDK reconstruction is mediocre (with respect to the other algorithms) as it has an SSIM (blue dot on the y-axis) of 0.925, which is the lowest of all after the iterative algorithms have hit 40 iterations. By increasing the number of projections to more than the 100 used here, we could improve the quality of the reconstruction.

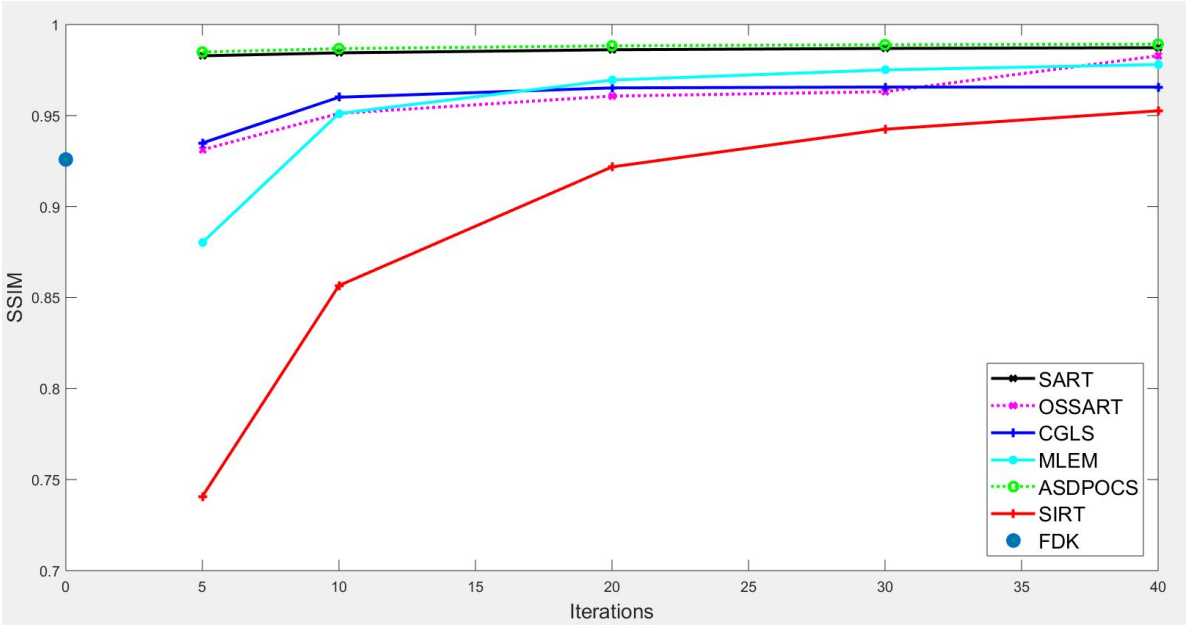


Figure 4.22: The SSIM for different number of iterations (reconstructions from the ideal projections)

Adding noise to the projections does not have a great influence on the SSIM of the different reconstructions. But we can see on Figure 4.23 that the SSIM of the SART and ASD-POCS reconstructions is decreasing with the number of iterations. This is only visible on those two algorithms as the other reconstructions techniques are improving with the number of iterations.

MLEM has no difficulty handling the noise and shows similar performances to the ones with the ideal projections. This confirms the fact that statistical algorithms are good at handling noise. Certainly if we compare this to the SSIM of the FDK reconstruction which has decreased

from 0.925 to 0.85 due to the addition of noise. As mentioned previously, this is one of the drawbacks of FDK: it struggles handling noise.

The SIRT reconstruction technique also handles the noise well, while OS-SART struggles a bit more as we can see on the Figure 4.23. This SSIM has not the final increase it has disclosed on Figure 4.22.

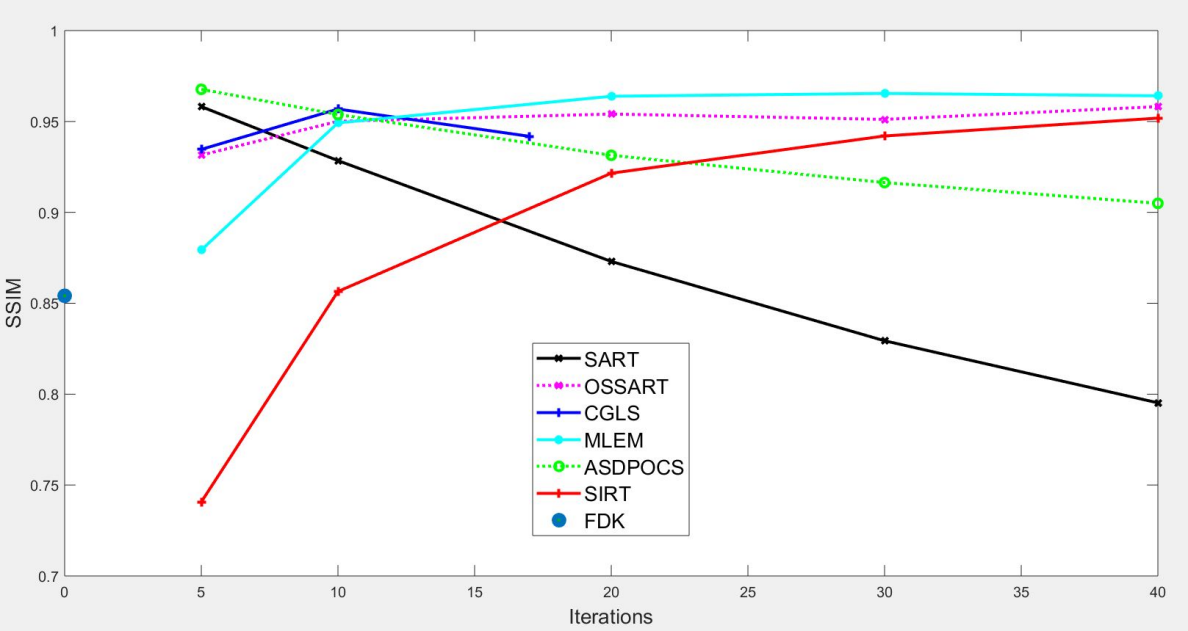


Figure 4.23: The SSIM for different number of iterations (reconstructions from the noisy projections)

Looking at the SSIM when varying the number of projections (Figures 4.24 and 4.25) the FDK algorithm is the most subjected to change due to the number of projections. For the iterative algorithms, the change in number of projections has little or no influence on the SSIM of the reconstructions with both noisy and ideal projections. As told before this is one of the reasons that iterative algorithms are so interesting in comparison with the classic FDK approach: less data gives similar results.

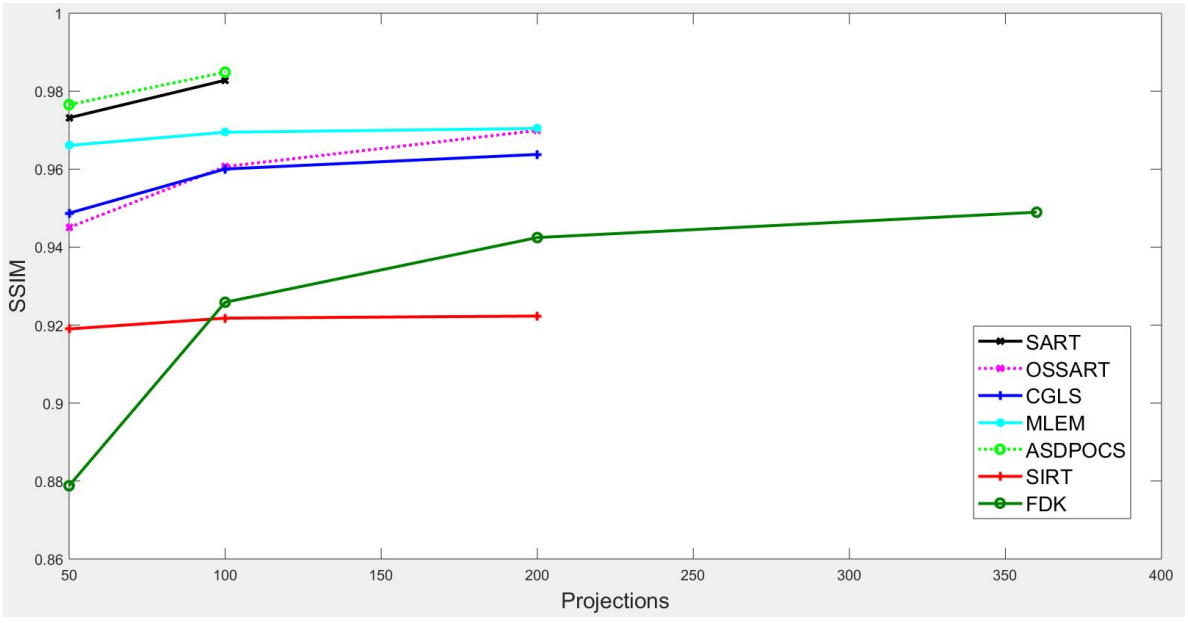


Figure 4.24: The SSIM for different number of (ideal) projections

As the trends of the SSIMs is similar for the ideal and the noisy case, we see a small decrease in SSIM due to the noisy projections. When improving the number of iterations a decrease in SSIM was noticeable (Figure 4.23) for the reconstructions with SART and ASD-POCS, the same trend is not visible with the number of projections. This will be discussed in Chapter 5.

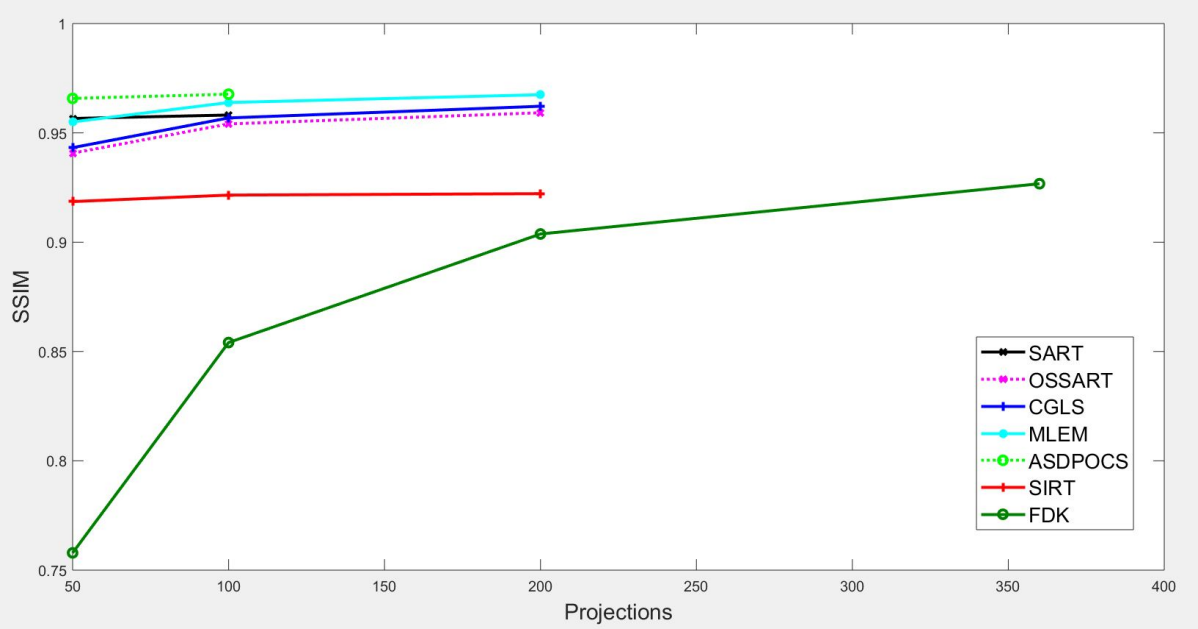


Figure 4.25: The SSIM of the different algorithms for different number of (noisy) projections

UQI is close to SSIM, as is explained in Section 3.4.1. The results obtained by UQI are similar to the ones presented here above and are displayed in Appendix A.2.2.

RMSE

Figures 4.26, 4.27, 4.28 and 4.29 show similar ranges for the Root Mean Square Error. In contrast with the other image quality metrics which all have values below 1, here the values are between 50 and 240. This is because the RMSE is expressed in HUs.

As in Section 4.2.3 SIRT is again surpassed by the other algorithms. The RMSE of the SIRT reconstructions only goes below 100 HU for 40 iterations. The RMSE of the OS-SART reconstruction has an analogous shape until 30 iterations but improves significantly going from 30 to 40 iterations.

MLEM and CGLS have RMSEs that are similar to OS-SART. With the sole difference that they do not improve after 30 iterations, as their SSIM is getting steady after 20 iterations.

SART and ASD-POCS are outperforming the other algorithms: no other algorithm is reaching the RMSE value they reach after 5 iterations. This is a huge contrast with FDK that has an RMSE of almost 100 HU more. Moreover the SART and ASD-POCS have a slight improvement of 10 HU as the iterations increase.

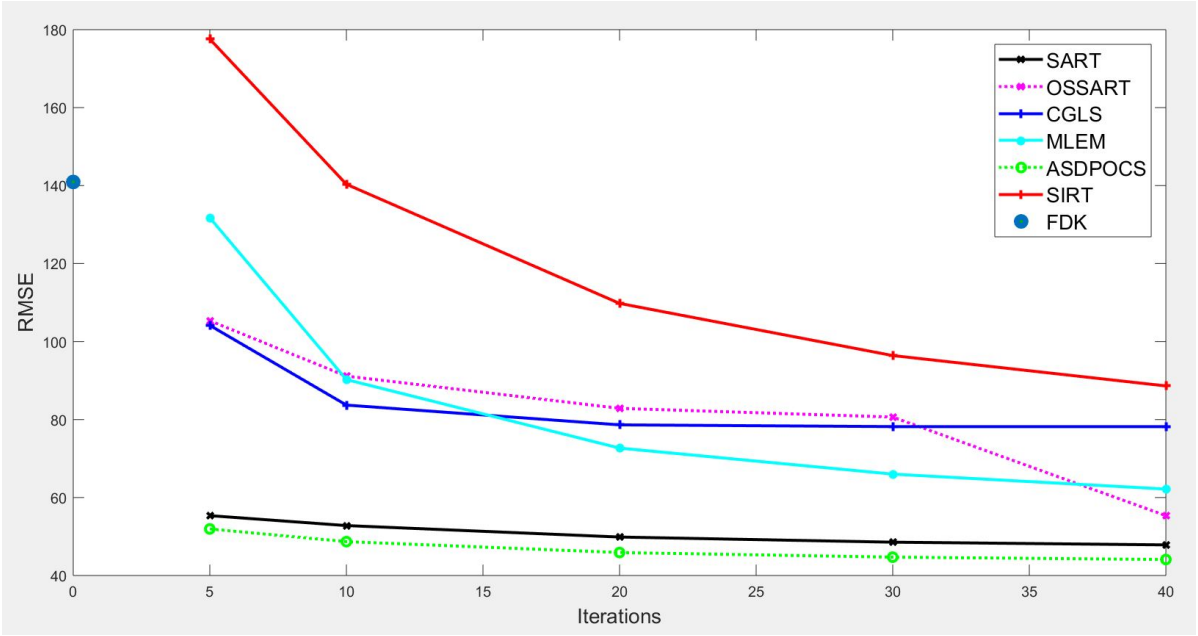


Figure 4.26: The RMSE for different number of iterations (reconstructions from the ideal projections)

The first observation on Figure 4.27, representing the RMSE of the reconstructions from noisy projections, is the soaring of the RMSE of SART: a rise of more than 100 HU. The same phenomenon is visible for the ASD-POCS algorithm, where the RMSE increases of 50 HU.

Overall the RMSE has increased, as the noise causes the reconstructions to be of less quality. For example the RMSE of the OS-SART reconstructions after 40 iterations was almost 50 HU in the case of the ideal reconstructions while for the reconstruction from noisy projections it is slightly above the 80 HU. The SIRT algorithm is the exception as the RMSE of both cases are similar.

FDK is again performing poorly when the noisy projections are used for reconstruction, as the RMSE is above 180. Using 100 projections seem not enough to handle the noise from these projections.

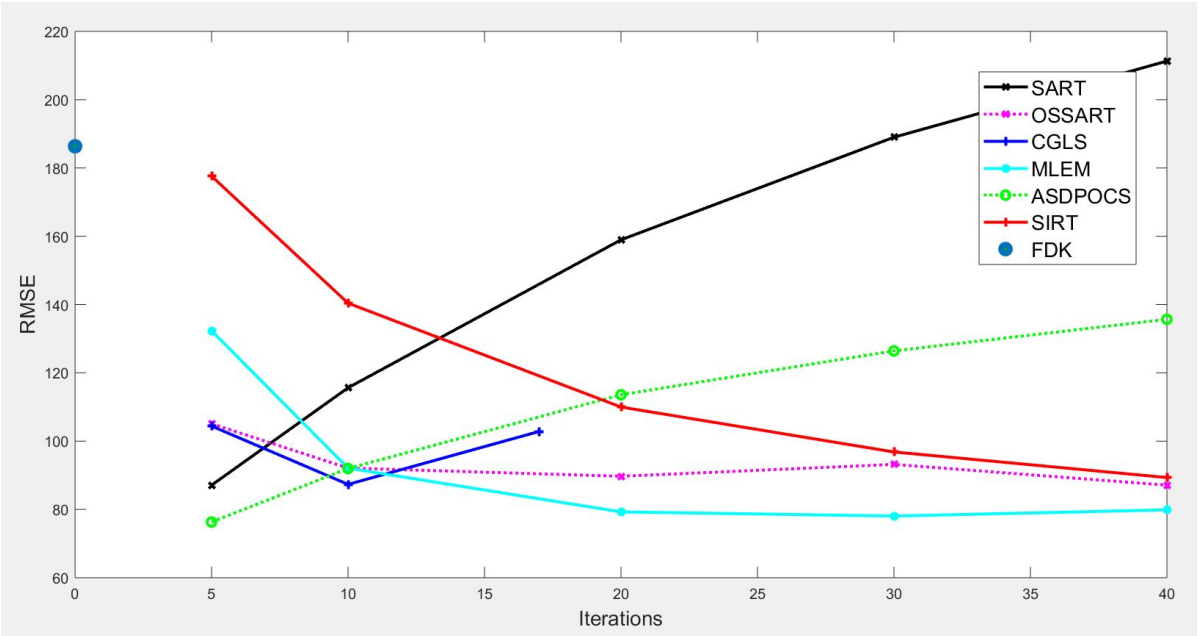


Figure 4.27: The RMSE for different number of iterations (reconstructions from the noisy projections)

The influence of the number of projections is only strongly visible for the FDK reconstructions. The iterative reconstruction are keeping a steady RMSE. This confirms what has been said in Section 4.2.3: the influence of the number of projections with respect to the image quality seems minimal.

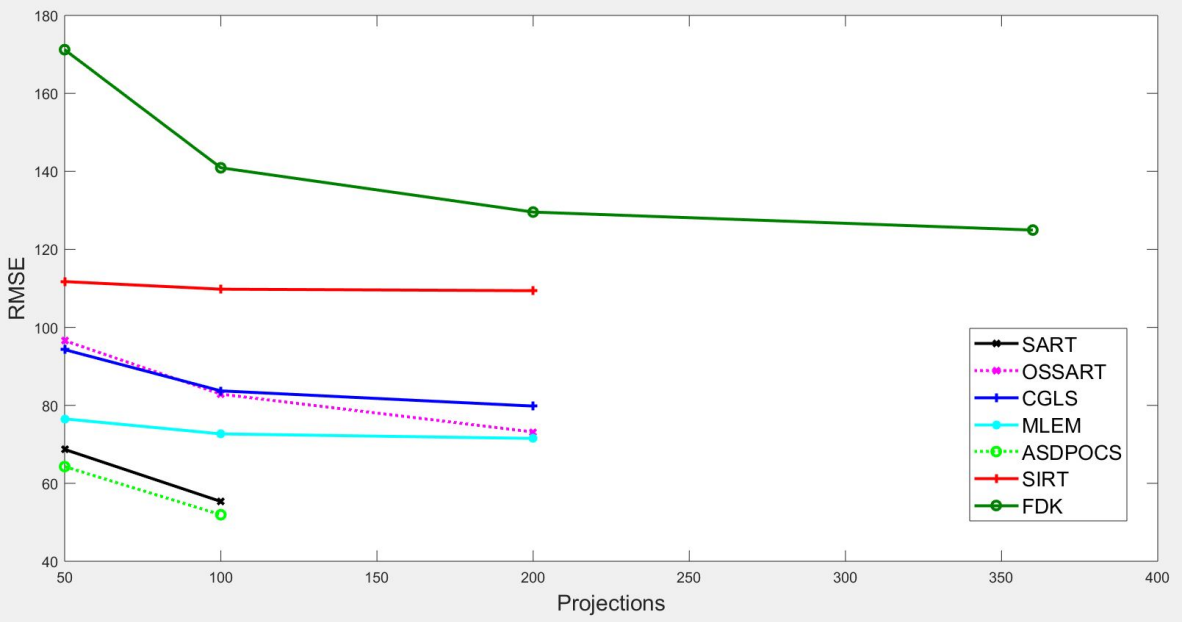


Figure 4.28: The RMSE of the different algorithms for different number of (ideal) projections

When the reconstruction is produced with noisy projections, increasing the number of projections influences the performance of the MLEM reconstructions. In contrast with the ideal projections, where SART and ASD-POCS are the best performing algorithms. The difference with the other reconstruction techniques is narrower in the case of noisy projections. MLEM even outperforms them when using 200 projections.

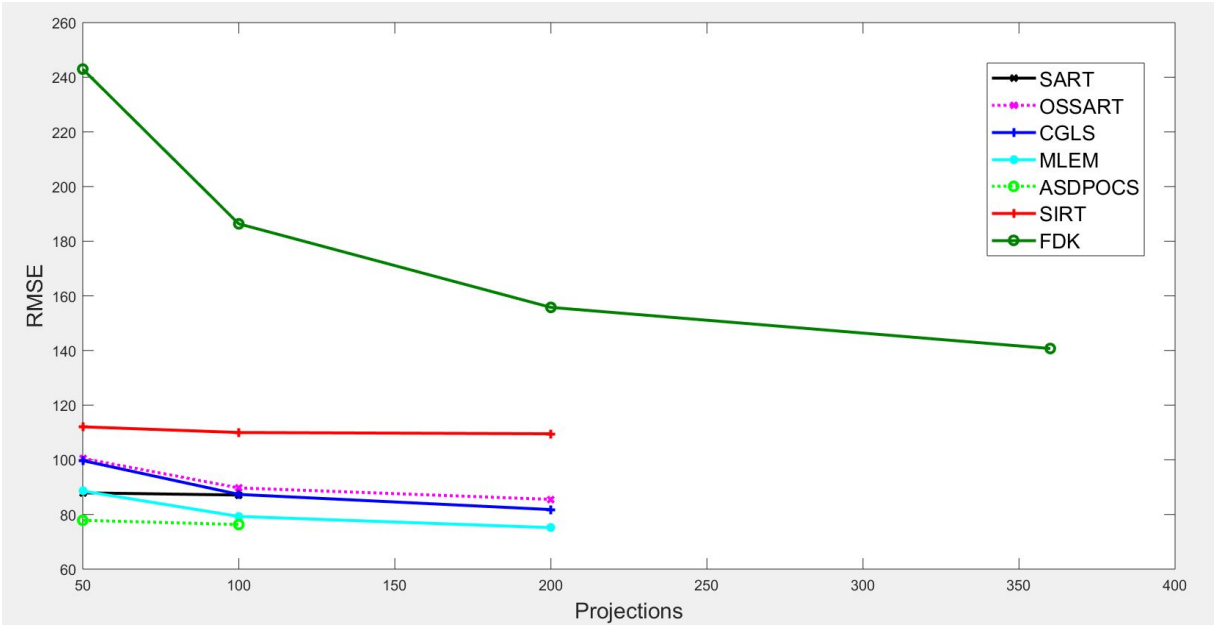


Figure 4.29: The RMSE of the different algorithms for different number of (noisy) projections

The RMSE confirms that MLEM is performing well, even with noisy projections, as for both SSIM and RMSE it has the best values with noisy projections.

Correlation Coefficient

The first thing that catches the eye are the values which are close to each other (as with SSIM): in Figure 4.30 the CC is between 0.88 and 0.98 and in Figure A.7 in Appendix A the CC varies between 0.79 and 0.96. The maximal value of the Correlation Coefficient is 1. The high values imply that the images have a high linear dependence with the reference reconstruction. This is what we are looking for.

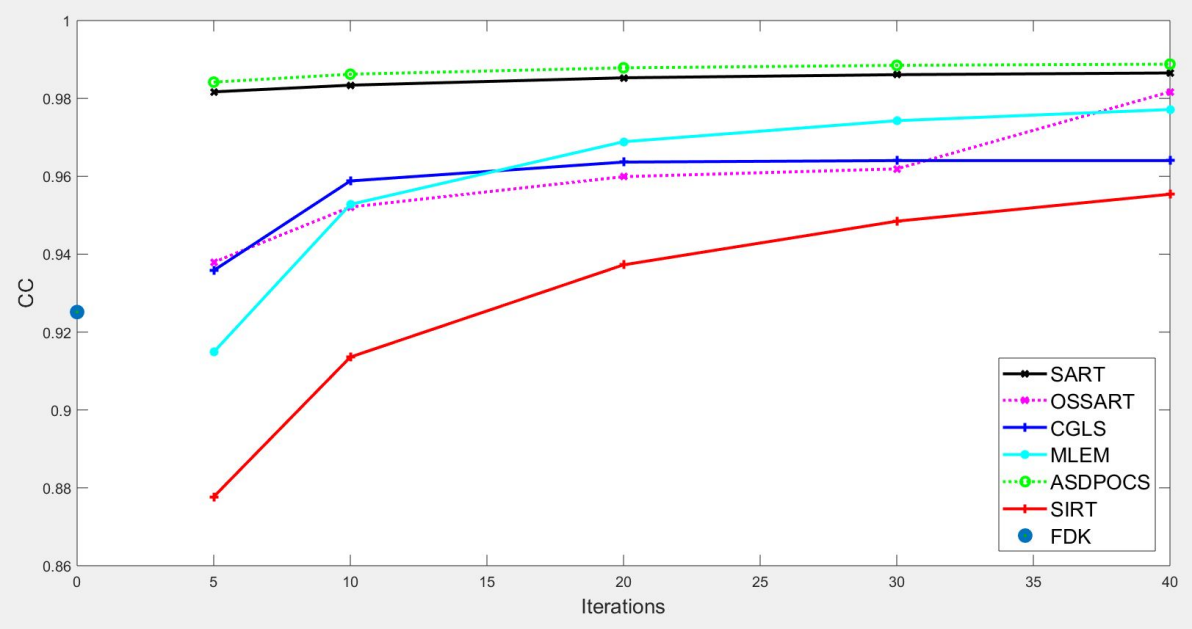


Figure 4.30: The CC for different number of iterations (reconstructions from the ideal projections)

Again, SART and ASD-POCS are performing well and SIRT is the algorithm generating the lowest CC, which is consistent with the images exhibited in Sections 4.2.1 and 4.2.2.

Adding noise to the projections show the same trends as for SSIM in Section 4.2.3. This can be seen in Appendix A on Figure A.7.

The linear dependence between the reference image and the reconstructions confirms the previous metrics. Increasing the number of projections makes, based on the CC criteria, the FDK reconstructions better than the SIRT ones. This matches better with the visual results than the other metrics were FDK is less performing than the SIRT algorithm. Moreovere, FDK is once more the only algorithm that is significantly affected by the number of projections.

With noisy projections, the CC of FDK is again below the one of SIRT (shown on Figure A.8 in Appendix A).

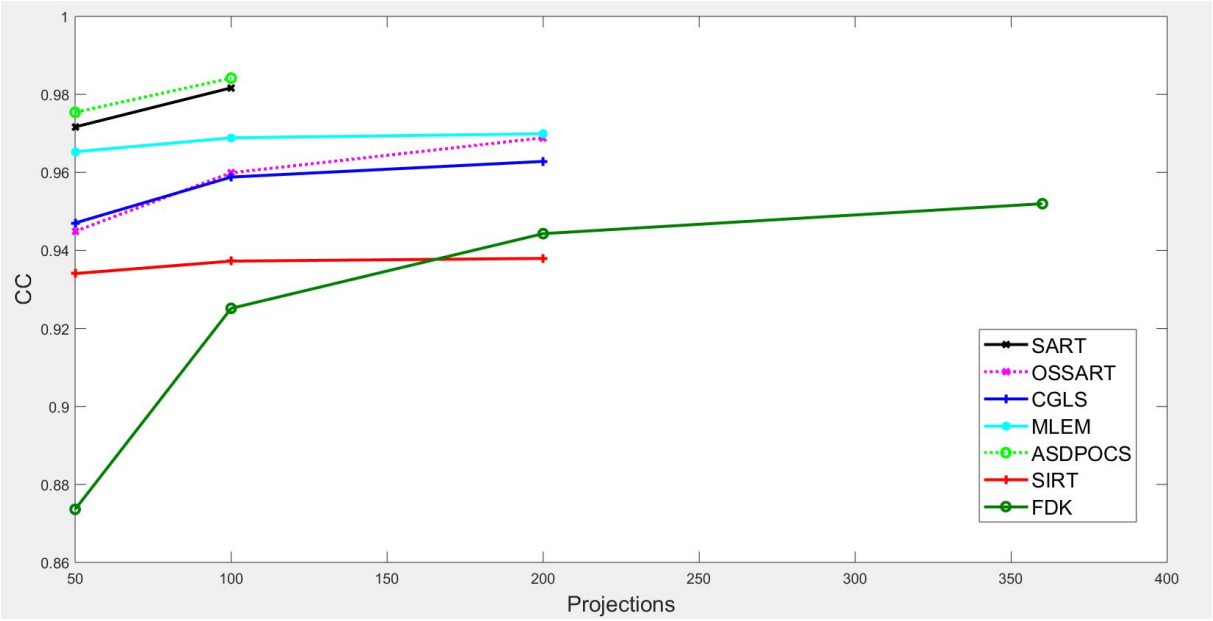


Figure 4.31: The CC of the different algorithms for different number of (ideal) projections

Running time

The time elapsed for the reconstructions is shown in the ideal case on Figures 4.32 and 4.33. On Figure 4.32 three groups can be distinguished: first SART, ASD-POCS and CGLS are one group. Second, SIRT, OSSART and MLEM are another one. Finally FDK is the last group.

The graph on Figure 4.32 shows why FDK is still the standard algorithm for CBCTs nowadays because the amount of time spared is significant. FDK, in the case of 100 projections, has a running time of 4 to 5 seconds. The fastest iterative algorithm after 5 iterations is SIRT with a running time of more than 35 seconds, but with results that visually and based on metrics are not even close to the ones obtained with FDK. The MLEM algorithm is the second fastest iterative algorithm. It shows some promising image quality after 10 to 20 iterations with a time consumption that, in comparison with other iterative algorithms, is still relatively low.

The global trend when increasing the number of iterations is a linear evolution. The only exception is the CGLS algorithm that after 20 iterations starts to use less time per iteration.

Globally, adding noise adds a bit of computation time to all the algorithms (Figure A.9 and A.10 in Appendix A). The only algorithm that is not affected at all is FDK, but the resulting quality of the image with noisy projections is not sufficient to be used for medical purpose. More projections are needed to give useful results with noisy projections.

When looking at the influence of the number of projections (Figure 4.33) we see the same linear evolution as with the number of iterations. With again FDK that outperforms all the other algorithms with more than 1 minute. For 200 projections FDK needs 9 seconds, while SIRT (the fastest iterative algorithm) needs more than 4 minutes to get a result that is (visually) not as good.

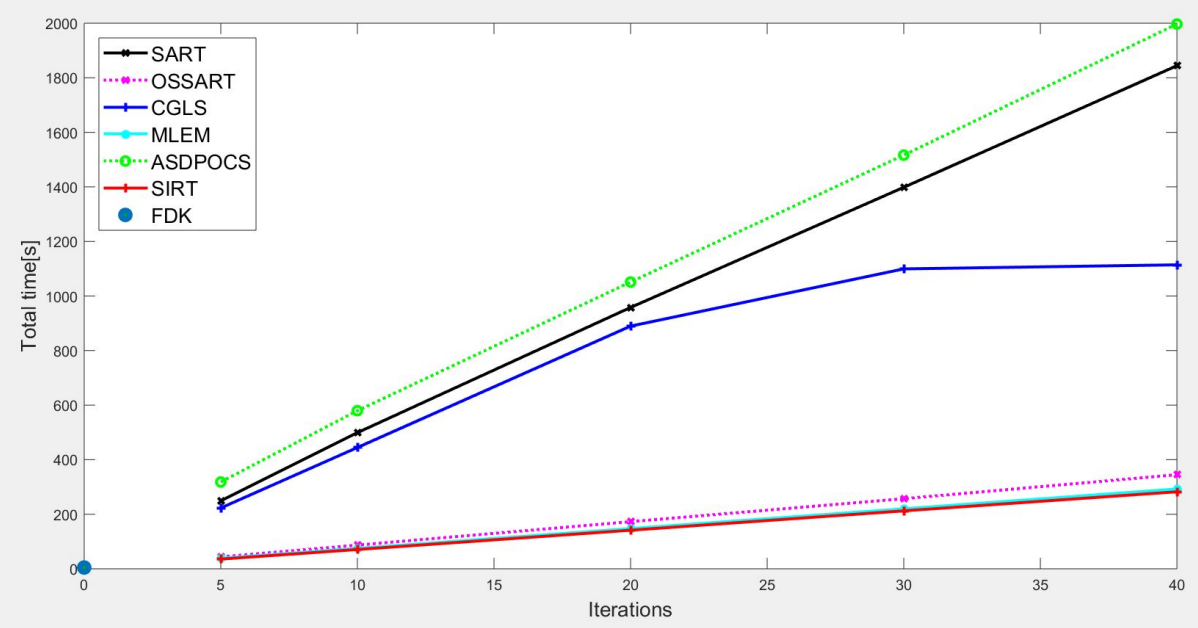


Figure 4.32: The time of the different algorithms for different number of iterations (reconstructions from the ideal projections)

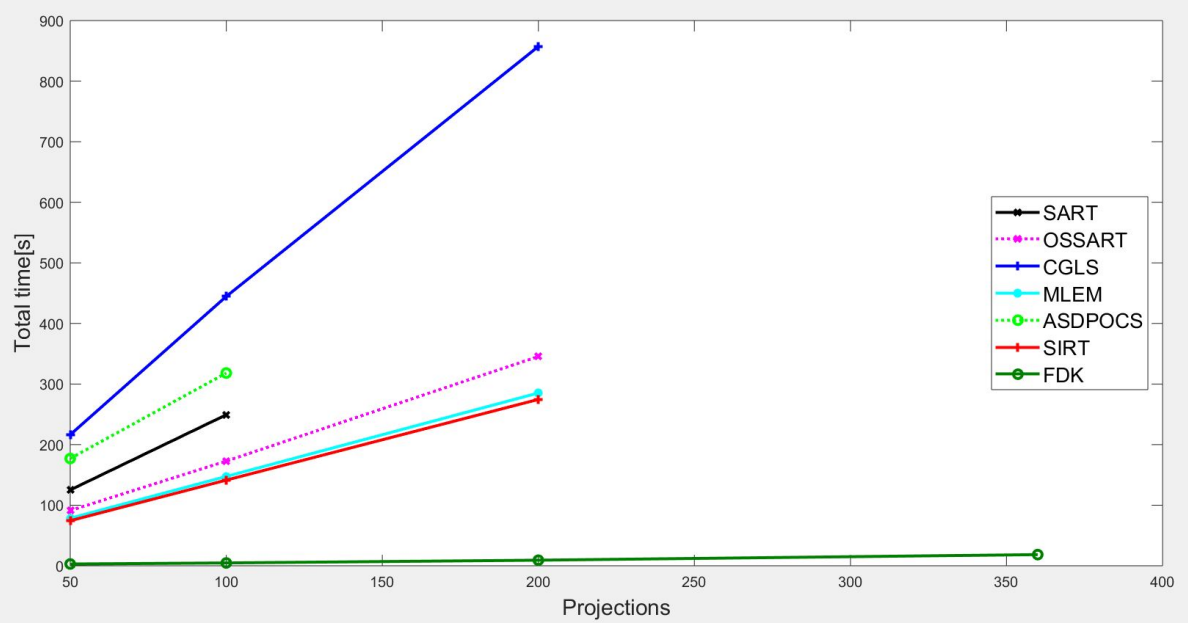


Figure 4.33: The time of the different algorithms for different number of (ideal) projections

Chapter 5

Discussion of the results

The previous chapter presented reconstructions and graphs with results of the evolution of the metrics. This chapter will focus on the discussion of these results and a more global comparison between all the reconstruction algorithms.

Spatial Resolution The first observation made in the results from images and metrics is the quality of the SART and ASD-POCS reconstructions. While with 5 or 10 iterations, most algorithms did not have good reconstructions, these algorithms were already thriving and showing some good results. The sharpness permits the details to be visible, this is illustrated by the close-up of Figure 5.1(a). This is a zoom on the reconstruction with 10 iterations of the SART algorithm. In comparison, the MLEM reconstruction is also shown, but this was produced with 40 iterations. In the figure below, we can see that Figure 5.1(c) does not show as much detail as the SART algorithm does with 10 iterations.

(a) SART: 10 iterations (ideal)

(b) SART: 10 iterations (noisy)

(c) MLEM: 40 iterations (ideal)

(d) MLEM: 40 iterations (noisy)

Figure 5.1: Reconstruction with MLEM for different number of projection angles

The details are visible with the dark spot in the middle of Figure 5.1(a), which is less sharper on Figure 5.1(c). The dark spot is barely visible on the reconstructions from the noisy projections, which tells us that noise has an effect on the spatial resolution: loss in sharpness.

Noise and artifacts Moreover, we can see that MLEM seems less subjected to noise than SART, which was also visible on the images and graphs in Section 4.2.1 and in Section 4.2.3. To explain the fact that SART and ASD-POCS are suffering more from noise than the other algorithms, two phenomena have to be looked at. First, there is the spatial resolution. As images are sharper and spatial resolution higher, it will also make noise (if not handled) and other artifacts more visible. So, there is a trade-off between spatial resolution and influence of unwanted effects such as noise and artifacts: the more iterations are used, the sharper the reconstructions, but also the more noise and the more artifacts tend to show up. This phenomenon is also visible for other algorithms such as OS-SART and CGLS, but not for all. The MLEM algorithm manages to keep the impact of noise under control. So in the case of MLEM, increasing the sharpness by more iterations has less influence on the collateral effects. Additionally, something remarkable was noticed with the reconstructions of the MLEM: the support seemed to have fade away.

Figure 5.2 shows the difference between the reconstruction of ASD-POCS and MLEM for 50 noisy projections. The support is not visible on the MLEM reconstruction, this is probably due to the fact that the MLEM reconstruction is less bright. It demonstrates that there are also some differences between the colour contrasts within the reconstructions. In this case this does not penalise the MLEM algorithm too much, as it still has a good spatial resolution.

(a) MLEM: 50 noisy projections

(b) ASD-POCS: 50 noisy projections

Figure 5.2: Reconstruction with ASD-POCS and MLEM for 50 noisy projections

For SART and ASD-POCS a second phenomenon which leads to the lowered metrics due to the increased number of iterations is overfitting. Figure 5.3 clearly shows that in case of reconstructions with noisy projections, the UQI of both algorithms decreases with the number of iterations increasing. SART is more affected than ASD-POCS. Overfitting means that the iterative methods are trying to incorporate the noise within the image, as they think it is a part of the image. This happens when the reconstruction methods have already converged and one keeps increasing the number of iterations. This is the case for ASD-POCS and SART as they already converge after 5 iterations. This is also the reason why the other algorithms do not suffer from overfitting: they did not converge yet. But if the number of iterations was increased further, the same phenomenon would be visible for all the iterative reconstruction techniques. Obviously, as FDK is not an iterative method, this does not apply to it.

Figures 5.2(a) and 5.2(b) do not show a lot of artifacts, apart from noise. This is true for most of the reconstructions shown in Chapter 4, as the only clearly visible artifacts are the white dot in the front part of the head and the line pattern that crosses the images in the case of less

satisfying reconstructions (for example the FDK reconstruction with 50 projections on Figure 4.15). This is due to the use of idealistic projections as explained in Section 3.2.2.

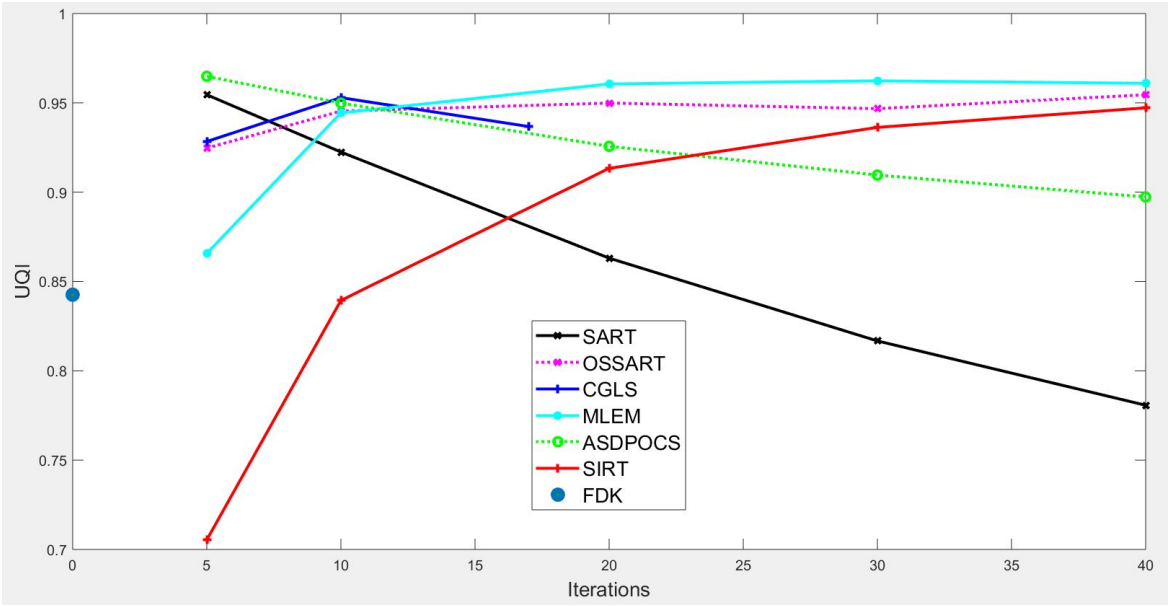


Figure 5.3: UQI: overfitting of SART and ASD-POCS

Data Dependence The overfitting phenomenon is specific to the number of iterations, and therefore not linked with the number of projections. The number of projections is the amount of data that is given to the algorithms. Theoretically, one would expect that the more data is given to an algorithm, the better the result would be, but this is not the case for the iterative algorithms. It even has little to no influence, if the amount of data is increased or decreased. As already explained, this is one of the main advantages of the iterative algorithms: they need less data to perform in comparison with FDK. This has been confirmed by the metrics, for example the correlation coefficients, of the different algorithms for the reconstructions with noisy projections on Figure 5.4. As FDK is improving significantly with the number of projections, the iterative algorithms are all keeping the same values of CC.

Figure 5.5(b) shows the influence of the number of projections to the image quality. FDK strongly depends on the number of projections: the more projections, the less noise is visible in the reconstruction. This does not only mean that a high number of projections has to be available, which is not always the case, but also that the patient is submitted to more radiation, which is something doctors try to avoid.

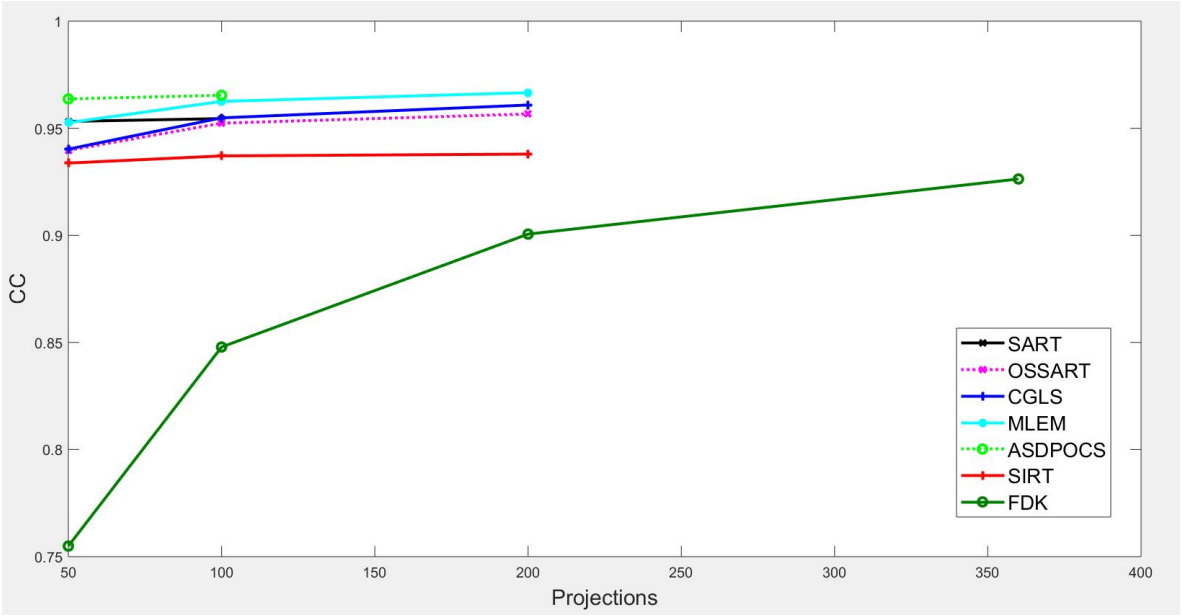


Figure 5.4: The CC of the different algorithms for different number of (noisy) projections

Limitations of metrics Metrics are very important when analysing and comparing images. But the quality assessment of a human observer who needs to use the image is at least as important. The majority of image quality metrics are putting the FDK reconstructions behind all the other algorithms, which does not correspond to the opinion of the observers. Figure 5.5 shows reconstructions with FDK and SIRT (20 iterations) with 200 projections. The metrics of SIRT are better but, looking at the images, SIRT is certainly not better than FDK. The FDK reconstruction looks a lot sharper. This is a limitation for metrics: they are useful as objective criterion but they are not sufficient to asses image quality on their own.

(a) SIRT: noisy projections

(b) FDK: noisy projections

Figure 5.5: Reconstruction with 200 projections

Time-Image Quality trade-off The highest possible image quality is clearly the main goal of the image reconstruction techniques. Low quality reconstructions are not useful at all for medical applications. But the time consumed by the algorithms is also something important. FDK is known for its speed, and this has been confirmed. Looking at the time performances, the other algorithms did not come near FDK. For 200 iterations, FDK needs less than 10 seconds with a more than decent result as outcome. The other algorithms show a trade-off between time and image quality. ASD-POCS and SART are visually and based on the metrics the best reconstruction techniques. But for 5 iterations both algorithms already need more than 6 minutes to reconstruct the images. CGLS is also time consuming, but with worse results in terms of

image quality.

Even if the number of projections does not have a big influence on the image quality, it does on the running time of the algorithms. Doubling the number of iterations and doubling the number of projections have the same effect: they double the running time. This is a remarkable result for iterative algorithms: it means that by halving the number of projections (from 100 to 50 for example), which does not change the image quality drastically, while doubling the number of iterations, the running time stays identical but the image quality will increase. This is the reason why iterative algorithms are so interesting in applications with less available data.

The trade-off between time and image quality finds its equilibrium with the OS-SART and MLEM algorithm. In comparison with SART, CGLS and ASD-POCS, they are much faster. Additionally, they produce better reconstructions with respect to SIRT and CGLS. Lastly, MLEM is robust against noise. From the iterative algorithms, MLEM seems to have the best compromise between time and image quality. SART and ASD-POCS are really close to each other and are the best reconstruction techniques with respect to image quality. However, the time they consume is too penalising to be a credible/useful alternative for FDK. SIRT is in opposition with those two algorithms: it is much faster but the image quality is too poor to produce useful images. CGLS was unsatisfactory. Its time requirements are not fully compensated by higher image quality.

Chapter 6

Conclusion

The main goal of this master thesis was to compare the statistical and iterative reconstruction techniques for CBCTs. Moreover, these have been compared with the FDK algorithm which is the standard reconstruction technique for CBCTs.

The comparison and discussion explained why FDK is still the reference reconstruction algorithm: good performances in less than 10 seconds. The influence of noise can be limited by multiplying the number of projections, which does not drastically augment the computation time. By increasing the number of projections however, the radiation exposition of the patient increases too. As we try to limit this exposition, iterative and statistical algorithms are used for the reconstruction. FDK is unable to retain the same quality when less data is available.

The image quality of the iterative algorithms was promising as it often matched the quality obtained with a good FDK reconstruction. However, this goes together with a substantial computation time. To obtain reconstruction results of similar quality to FDK, the algorithms had to run for at least 10 times longer. The trade-off of time for image quality can be acceptable in certain contexts, but is not always realistic.

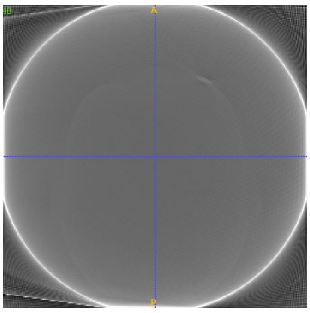
All in all, we can say that FDK is still the best reconstruction algorithm in applications where speed is crucial. But if we focus on the reconstructions from less data, which means less radiation to the patient and a less expensive procedure, alternative algorithms such as MLEM give promising results combining both image quality and reasonable time delays.

Appendix A

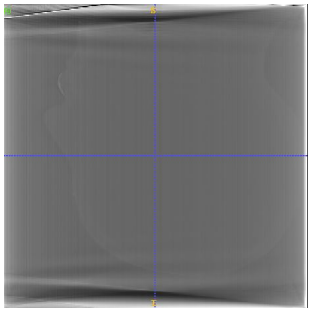
Reconstructions and metrics

Some reconstructions and metrics were not showed in the core text of the master thesis but will be shown in the following sections.

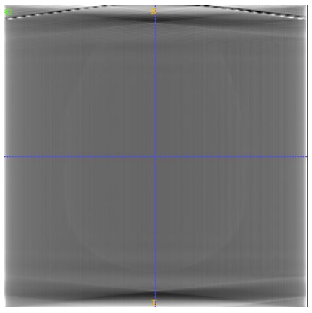
A.1 Reconstructions from projections



(a) Transverse plane



(b) Sagittal plane

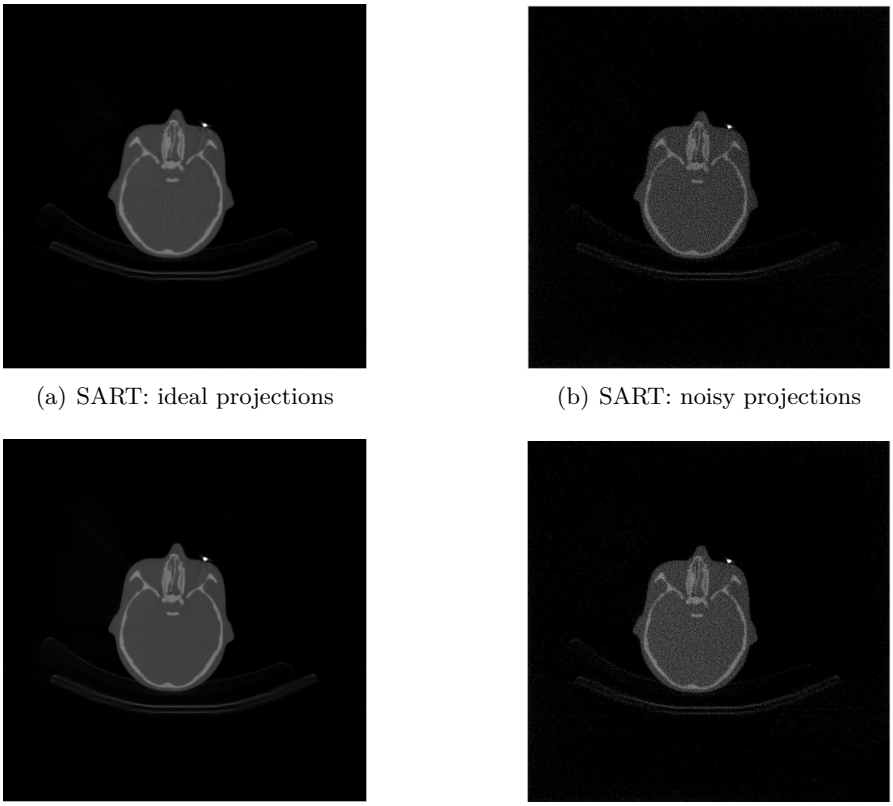


(c) Coronal plane

Figure A.1: Reconstruction with the OSSART algorithm (40 iterations) from HIS file projections

A.2 Reconstruction from projected reconstructions

A.2.1 Reconstructions with 10 iterations



(a) SART: ideal projections

(b) SART: noisy projections

(c) ASD-POCS: ideal projections

(d) ASD-POCS: noisy projections

Figure A.2: Reconstruction with SART and ASD-POCS algorithm: 10 iterations

A.2.2 Metrics

UQI

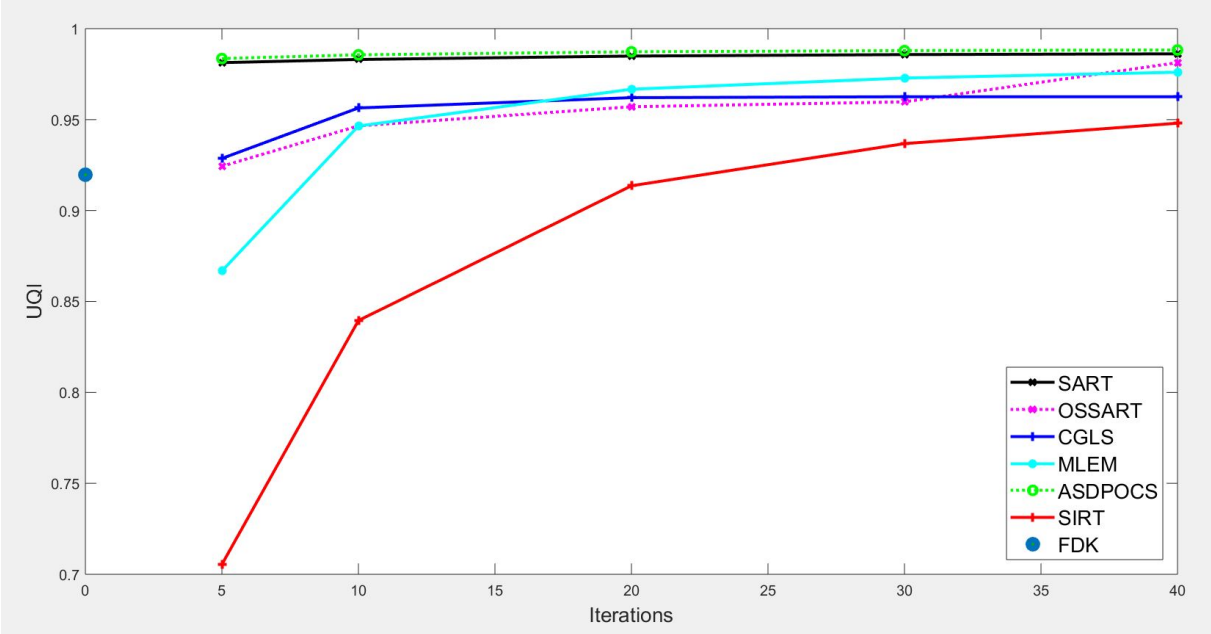


Figure A.3: The UQI of the different algorithms for different number of iterations (reconstructions from the ideal projections)

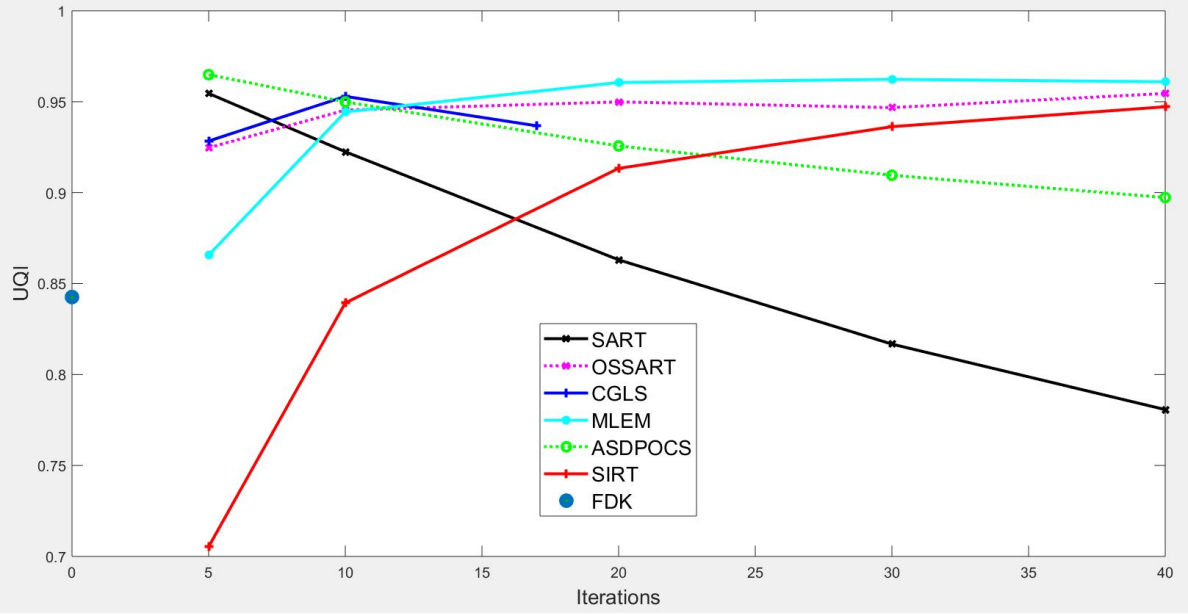


Figure A.4: The UQI of the different algorithms for different number of iterations (reconstructions from the noisy projections)

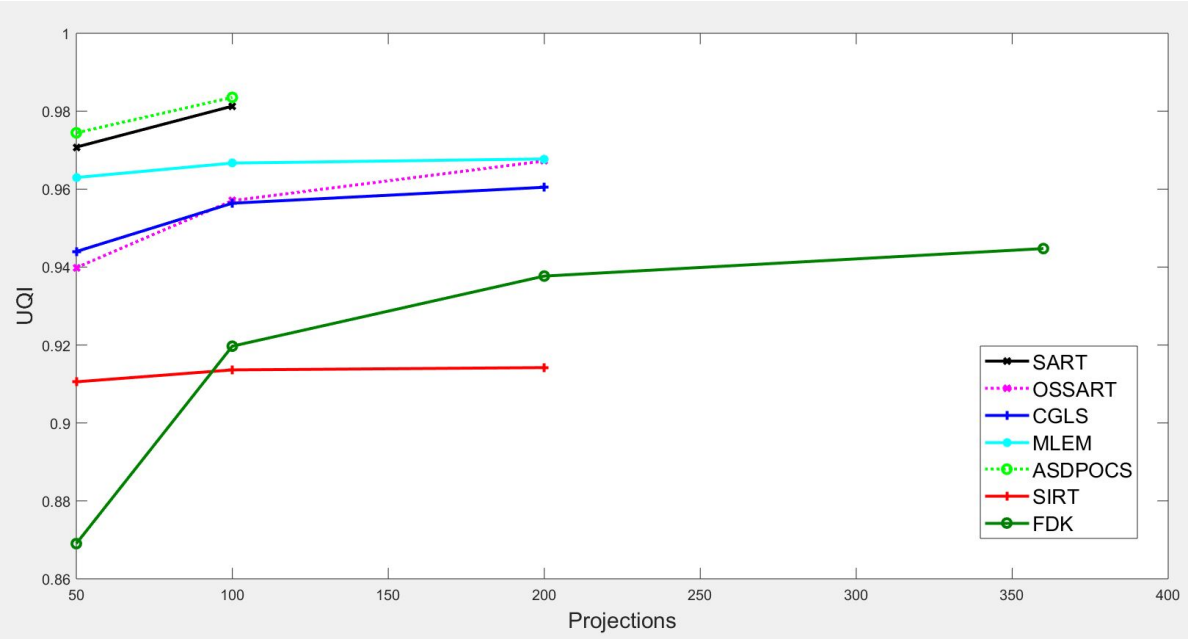


Figure A.5: The UQI of the different algorithms for different number of (ideal) projections

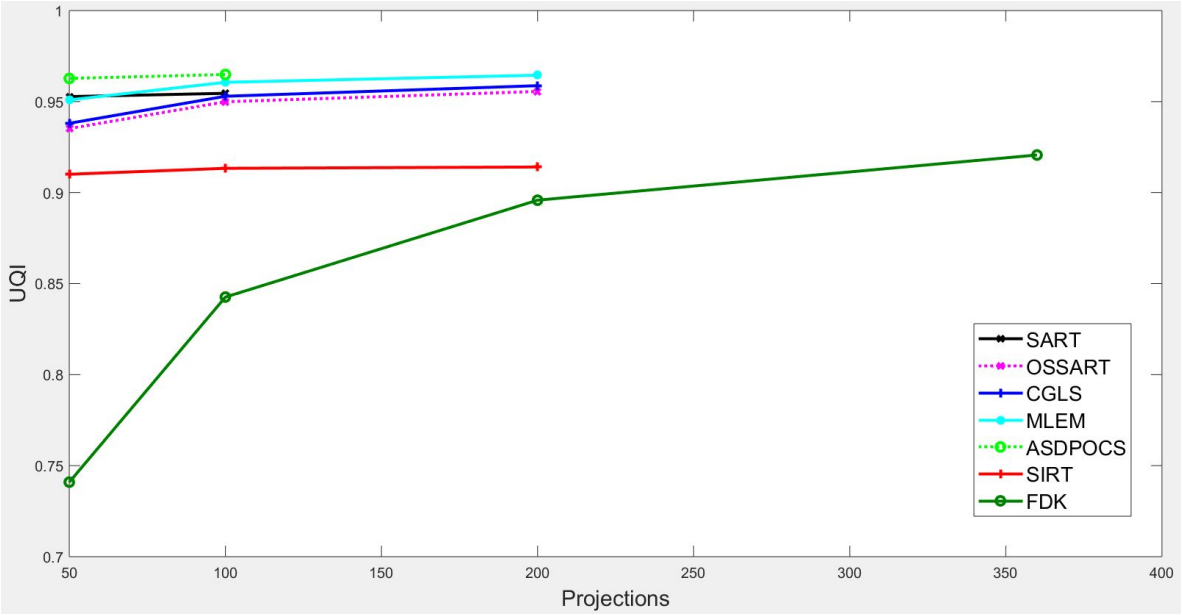


Figure A.6: The UQI of the different algorithms for different number of (noisy) projections

Correlation Coefficient

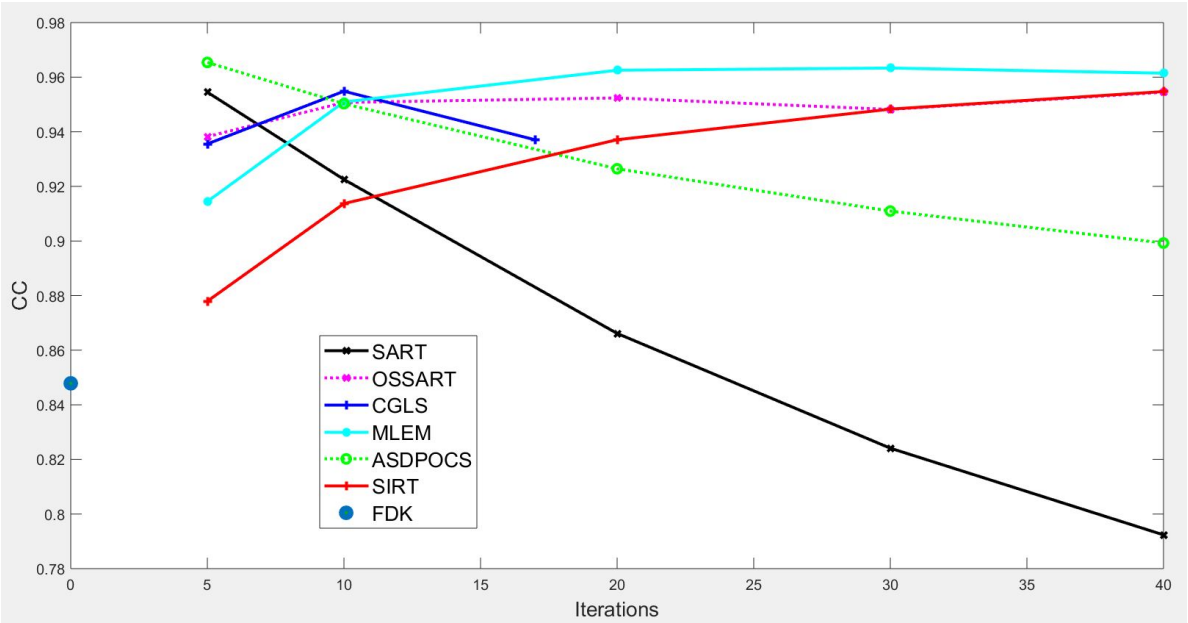


Figure A.7: The CC of the different algorithms for different number of iterations (reconstructions from the noisy projections)

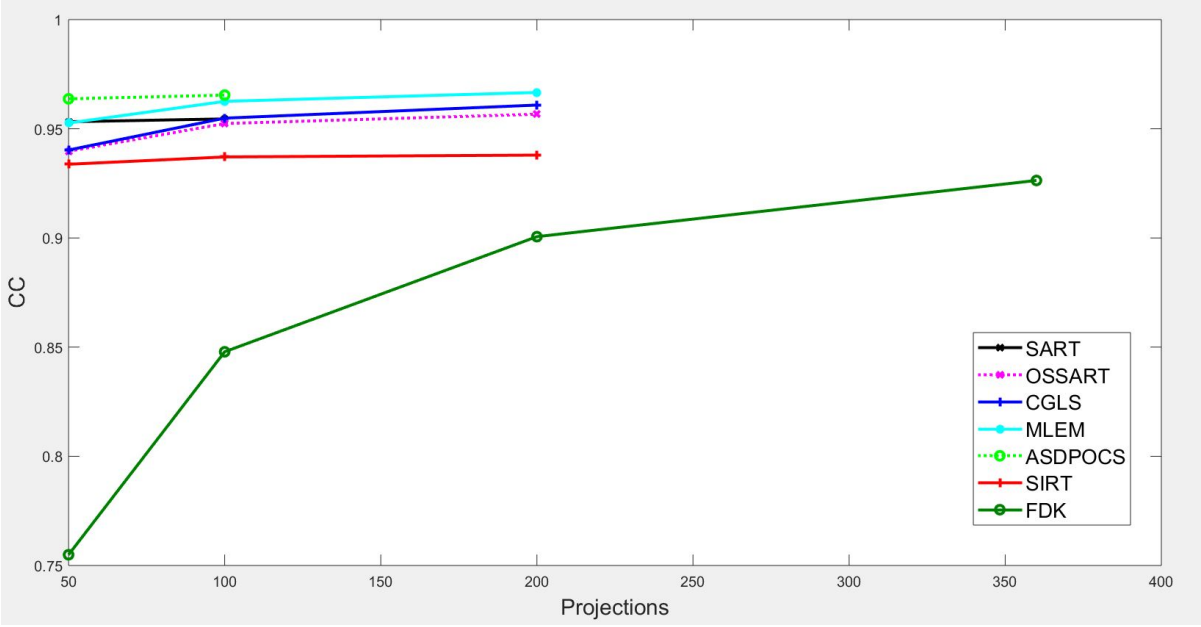


Figure A.8: The CC of the different algorithms for different number of (noisy) projections

Computation time

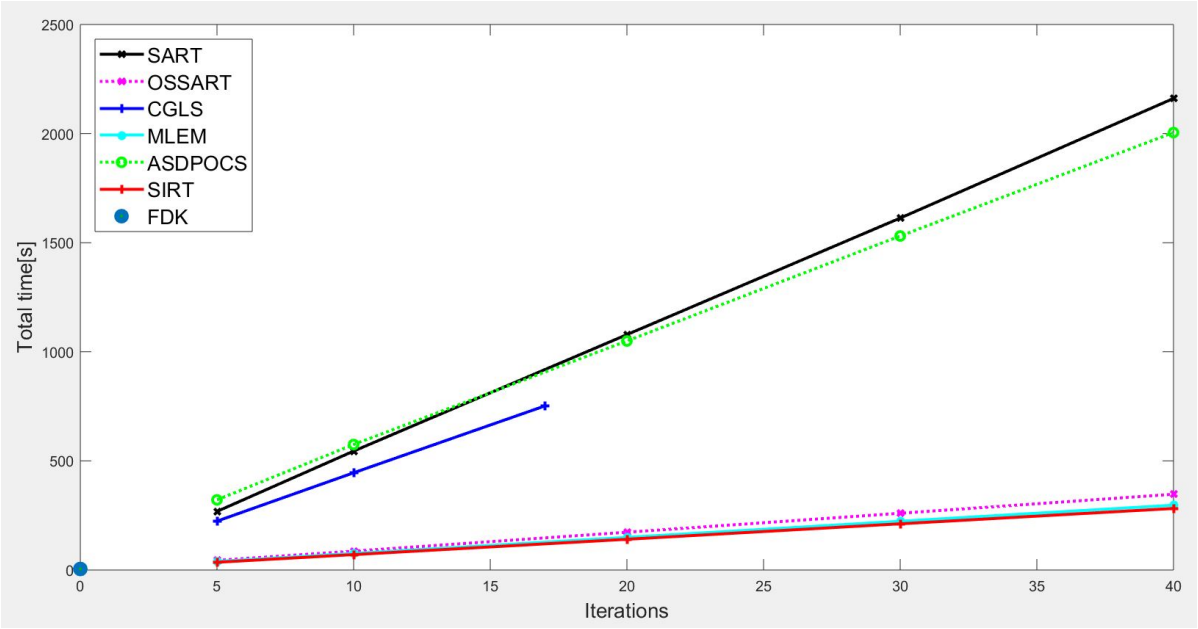


Figure A.9: The time of the different algorithms for different number of iterations (reconstructions from the noisy projections)

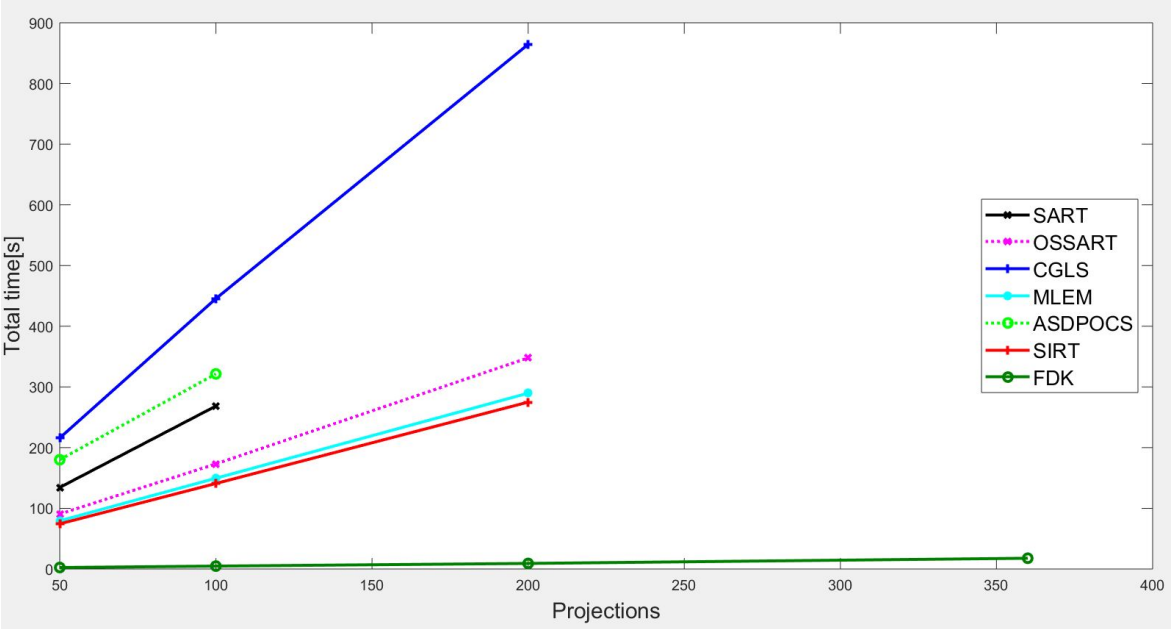


Figure A.10: The time of the different algorithms for different number of (noisy) projections

Metrics: summary table

The two first tables show the metrics of the reconstructions with varying the number of iterations and 100 projections (ideal and noisy).

Algorithm	# iterations	UQI	RMSE [HU]	CC	SSIM	time [s]
FDK	/	0.91969925	140.92911	0.92514181	0.92585295	4.666215
SART	5	0.98130310	55.352165	0.98168719	0.98281509	249.214977
SART	10	0.98305595	52.801346	0.98339969	0.98442054	499.010168
SART	20	0.98493034	49.863235	0.98529845	0.98614037	957.989280
SART	30	0.98573601	48.544399	0.98609978	0.98687983	1398.529382
SART	40	0.98614901	47.850563	0.98653358	0.98725897	1844.389708
OS-SART	5	0.92433172	105.33923	0.93793696	0.93120003	43.534196
OS-SART	10	0.94652504	91.131844	0.95209646	0.95114177	86.834759
OS-SART	20	0.95705545	82.871109	0.95991021	0.96066755	172.790709
0S-SART	30	0.95975244	80.634315	0.96190554	0.96310753	256. 997488
OS-SART	40	0.98130310	55.352165	0.98168719	0.98281509	344.974777
SIRT	5	0.70544666	177.57545	0.87769514	0.74077654	35.500420
SIRT	10	0.83949995	140.34769	0.91357589	0.85658115	70.727525
SIRT	20	0.91361094	109.78868	0.93725926	0.92178428	141.374689
SIRT	30	0.93677962	96.410660	0.94846714	0.94247162	212.398740
SIRT	40	0.94800627	88.618599	0.95539153	0.95256829	282.352987
CGLS	5	0.92864048	104.14557	0.93581992	0.93488765	222.866608
CGLS	10	0.95639449	83.710861	0.95881146	0.93488765	444.554933
CGLS	20	0.96206027	78.634293	0.96364880	0.96004266	889.581745
CGLS	30	0.96256381	78.187805	0.96403825	0.96519667	1099.403148
CGLS	40	0.96256381	78.187805	0.96403825	0.96565348	1114.336768
ASD-POCS	5	0.98355395	51.939720	0.98418689	0.98488188	318.150238
ASD-POCS	10	0.98560721	48.687065	0.98621857	0.98676455	579.499114
ASD-POCS	20	0.98726398	45.886753	0.98785746	0.98828441	1051.201815
ASD-POCS	30	0.98793030	44.706215	0.98849285	0.98889595	1516.487186
ASD-POCS	40	0.98826575	44.100849	0.98880458	0.98920381	1996.886449
MLEM	5	0.86677492	131.74678	0.91486740	0.88025892	38.529012
MLEM	10	0.94647062	90.248276	0.95277864	0.95113349	74.991090
MLEM	20	0.96669066	72.674431	0.96887606	0.96947771	147.389171
MLEM	30	0.97283512	65.999374	0.97428125	0.97508311	219.826052
MLEM	40	0.97604674	62.165535	0.97714734	0.97801667	292.535753

Table A.1: The metrics obtained with the ideal projections

Algorithm	# iterations	UQI	RMSE [HU]	CC	SSIM	time [s]
FDK	/	0.84260035	186.35513	0.84786594	0.85409158	4.760616
SART	5	0.95453519	87.100998	0.95451307	0.95816016	268.449711
SART	10	0.92247027	115.63020	0.92253327	0.92845929	544.834392
SART	20	0.86307156	158.96233	0.86613625	0.87299830	1078.783756
SART	30	0.81673414	189.00562	0.82411021	0.82935578	1612.820365
SART	40	0.78073144	211.30476	0.79236269	0.79522920	2161.054654
OS-SART	5	0.92468643	105.09410	0.93823016	0.93152064	44.171410
OS-SART	10	0.94537622	92.109818	0.95074207	0.95008528	86.485412
OS-SART	20	0.94989210	89.710091	0.95238799	0.95408785	173.142611
0S-SART	30	0.94674850	93.185860	0.94819289	0.95114046	260.043125
OS-SART	40	0.95453519	87.100998	0.95451307	0.95816016	347.224544
SIRT	5	0.70536286	177.60281	0.87794358	0.74071050	35.432123
SIRT	10	0.83938867	140.40901	0.91370225	0.85648662	70.604573
SIRT	20	0.91333979	109.98795	0.93709338	0.92154080	140.859170
SIRT	30	0.93627691	96.828705	0.94830489	0.94201523	211.770985
SIRT	40	0.94720984	89.335594	0.95472938	0.95184255	282.100895
CGLS	5	0.92837715	104.41464	0.93554986	0.93464482	223.925140
CGLS	10	0.95291597	87.332169	0.95485729	0.95682847	445.495054
CGLS	17	0.93675721	102.83198	0.93706357	0.94183075	752.035802
ASD-POCS	5	0.96487457	76.304565	0.96539968	0.96768957	321.438376
ASD-POCS	10	0.94989079	91.982780	0.95028573	0.95383656	575.126702
ASD-POCS	20	0.92567861	113.58568	0.92644966	0.931379682	1049.533527
ASD-POCS	30	0.90957379	126.43562	0.91095698	0.91639251	1530.701046
ASD-POCS	40	0.89731759	135.67517	0.89925653	0.90496022	2004.722641
MLEM	5	0.86569083	132.30003	0.91445595	0.87930858	39.158704
MLEM	10	0.94436729	92.090080	0.95093066	0.94921607	76.000492
MLEM	20	0.96057552	79.277351	0.96253979	0.96386552	149.507544
MLEM	30	0.96232402	78.059914	0.96333808	0.96542394	223.375857
MLEM	40	0.96092916	79.872215	0.96145111	0.96411359	297.049795

Table A.2: The metrics obtained with the noisy projections

The two following tables are the metrics with varying number of projections. The number of iterations is not the same for all the algorithms. The metrics of CGLS are produced with 10 iterations, SART and ASD-POCS with 5 iterations and OS-SART, SIRT and MLEM with 20 iterations.

Algorithm	# projections	UQI	RMSE [HU]	CC	SSIM	time [s]
FDK	50	0.86903393	171.21188	0.87361079	0.87878495	2.923725
FDK	200	0.93769270	129.56485	0.94430041	0.94246095	9.202965
FDK	360	0.94475120	124.94993	0.95197821	0.94896197	18.410464
SART	50	0.97079402	68.700005	0.97166675	0.97319043	125.127464
OS-SART	50	0.93987584	96.603882	0.94490689	0.94504827	91.132934
OS-SART	200	0.96723217	73.157509	0.96888822	0.96994346	345.445514
SIRT	50	0.91055185	111.73450	0.93407017	0.91901255	74.375582
SIRT	200	0.91420168	109.39764	0.93795687	0.92232168	274.588403
CGLS	50	0.94398904	94.282921	0.94703788	0.94871408	216.266468
CGLS	200	0.96048355	79.806213	0.96282315	0.96378595	857.200766
ASD-POCS	50	0.97445685	64.285614	0.97541982	0.97654891	176.999798
MLEM	50	0.96299070	76.529457	0.96531403	0.96609408	78.453650
MLEM	200	0.96776140	71.510124	0.96991509	0.97045767	285.188616

Table A.3: The metrics obtained with the ideal projections

Algorithm	# projections	UQI	RMSE [HU]	CC	SSIM	time [s]
FDK	50	0.74093789	242.98549	0.75491625	0.75790215	2.430903
FDK	200	0.89583349	155.77840	0.90058595	0.90374732	9.145174
FDK	360	0.92065179	140.72420	0.92632157	0.92673582	17.708785
SART	50	0.95273304	87.903084	0.95319843	0.95658129	134.393478
OS-SART	50	0.93517691	100.53387	0.93968767	0.94072920	90.544423
OS-SART	200	0.95555919	85.513351	0.95666575	0.95920044	347.883372
SIRT	50	0.91007400	112.06563	0.93374366	0.91858119	74.423369
SIRT	200	0.91404551	109.51906	0.93793654	0.92218310	274.917697
CGLS	50	0.93806511	99.708618	0.94026053	0.94323510	215.928545
CGLS	200	0.95872849	81.784187	0.96084380	0.96216506	864.407530
ASD-POCS	50	0.96271455	77.862228	0.96370810	0.96576017	179.983072
MLEM	50	0.95091414	88.581032	0.95260739	0.95500100	79.350606
MLEM	200	0.96448606	75.175255	0.96662420	0.96745443	289.652829

Table A.4: The metrics obtained with the ideal projections

Appendix B

MATLAB Scripts and Functions

The main part of the code is from the TIGRE toolbox and is available on the following link: `https://github.com/CERN/TIGRE`.

But some scripts and functions were needed to handle the used data sets. These functions and scripts will be described in the sections below.

B.1 Geometry structure

The geometry of CBCTs is a part that is needed for functions such as Ax and Atb that are the two main building blocks of the Tigre Toolbox. An example of a geometry is shown here below.

```
%the geometry (here the standard tigre geo)
% VARIABLE                                          DESCRIPTION
                        UNITS
%
-----------------------------------------------------------------------------------------

geo.DSD = 1536;                                     % Distance Source
    Detector       (mm)
geo.DSO = 1000;                                     % Distance Source Origin
            (mm)
% Detector parameters
geo.nDetector=[512; 512];                                                   %
    number of pixels                (px)
geo.dDetector=[0.8; 0.8];                                                   %
    size of each pixel              (mm)
geo.sDetector=geo.nDetector.*geo.dDetector; % total size of the
    detector     (mm)
% Image parameters
geo.nVoxel=[512;512;144];                           % number of voxels
                (vx)
geo.sVoxel=[512;512;144];                           % total size of the image
            (mm)
geo.dVoxel=geo.sVoxel./geo.nVoxel;                  % size of each voxel
                (mm)
% Offsets
geo.offOrigin =[0;0;0];                             % Offset of image from
    origin    (mm)
geo.offDetector=[0; 0];                             % Offset of Detector
                (mm)
```

```
17
18
19 % Auxiliary
20 geo.accuracy=0.5;                                % Accuracy of FWD proj
                (vx/sample)
```

B.2 HIS projections

B.2.1 HIS files handling function [9]

This function [9] transforms the projections, in the format of HIS files, within a 3D matrix.

```
1  function im = readHIS()
2
3  %Inspired from the readHIS function from Nick Rawluk
4  %Modified so it can transform a series of his files into a 3D matrix
5
6  fileLocation =
7  'C:\Users\alarnould\Documents\Alexandre\ImagXEcoReconstruction\
       Projections\20180322_JoeHead_CBCT_Elekta\img_1
       .3.46.423632.141000201832216454478l.119\00001.1.3.46.423632.14100020183221645
       his';
8  i = 1;
9  im = zeros(512, 512, 196);
10
11 while i<=196
12      fid = fopen(fileLocation);
13      if fid == -1
14          warning('File could not be opened');
15          im1 = -1;
16      else
17          % Read the file Header
18          header = fread(fid,50,'uint16')';
19
20          % Parse the header for data (only the dimensions currently)
21          Dim = header(9:10);
22
23          % Read and reshape the data.
24          im1 = uint16(fread(fid,Dim(1)*Dim(2),'uint16'));
25          im1 = reshape(im1,Dim)';
26          size(im1);
27          im(:,:,i) = im1;
28
29          fclose(fid);
30      end
31        i=i+1;
32       fileLocation = ['C:\Users\alarnould\Documents\Alexandre\
           ImagXEcoReconstruction\Projections\20180322
           _JoeHead_CBCT_Elekta\img_1
           .3.46.423632.141000201832216454478l.119\00' num2str(i,'%03d')
            '.1.3.46.423632.141000201832216454478l.119.his'];
```

```
end
im = single(im);
end
```

B.2.2 Angles extraction

As shown on figure 3.2, after having created the 3D matrix we also need the projection angles. They are extracted from a XML file with the function shown below.

```
function [angles, ucentre, vcentre] = extraction()

%open files
file = fopen('C:\Users\alarnould\Documents\Alexandre\
    ImagXEcoReconstruction\Projections\20180322_JoeHead_CBCT_Elekta\
    Param_img\_Frames.xml');
tline = fgetl(file);

%Initialization
angles = [];
ucentre = [];
vcentre = [];

%search paterns
pattern1_beg = '<GantryAngle>';
pattern2_beg = '<UCentre>';
pattern3_beg = '<VCentre>';

while ischar(tline)

    find1_beg = findstr(tline, pattern1_beg);
    find2_beg = findstr(tline, pattern2_beg);
    find3_beg = findstr(tline, pattern3_beg);

    if ~isempty(find1_beg)
        new_str1 = split(tline, '>');
        new_str2 = split(new_str1(2), '<');
        new_angle = str2double(new_str2(1));
        angles = [angles [new_angle]];
    end

    if ~isempty(find2_beg)
        new_str1 = split(tline, '>');
        new_str2 = split(new_str1(2), '<');
        new_ucentre = str2double(new_str2(1));
        ucentre = [ucentre [new_ucentre]];
    end

    if ~isempty(find3_beg)
        new_str1 = split(tline, '>');
        new_str2 = split(new_str1(2), '<');
        new_vcentre = str2double(new_str2(1));
        vcentre = [vcentre [new_vcentre]];
```

```
42      end
43
44      tline = fgetl(file);
45
46  end
47
48  fclose(file);
49
50  end
```

B.2.3 Reconstruction script

```
1  % Image Reconstruction Script for ImagX.Eco (Tigre toolbox)
2
3  % /!\ Each time you restart MATLAB and TIGRE, you must run the
       initTigre
4  % file from the Tigre-master\MATLAB directory.
5
6
7  addpath('C:\Users\alarnould\Documents\Alexandre\TIGRE-master\MATLAB')
       ;
8
9  Projections = readHIS();
10 %class(Projections)
11 nProj = size(Projections, 3); % Number of projections
12
13 %% Geometry
14
15 geo.DSD = 1536;                                % Distance Source
       Detector      (mm)
16 geo.DSO = 1000;                                % Distance Source Origin
           (mm)
17 % Detector parameters
18 geo.nDetector=[512; 512];                                        %
       number of pixels              (px)
19 geo.dDetector=[0.8; 0.8];                                       %
       size of each pixel            (mm)
20 geo.sDetector=geo.nDetector.*geo.dDetector; % total size of the
       detector     (mm)
21 % Image parameters
22 geo.nVoxel=[256;256;256];                    % number of voxels
                  (vx)
23 geo.sVoxel=[256;256;256];                    % total size of the image
           (mm)
24 geo.dVoxel=geo.sVoxel./geo.nVoxel;          % size of each voxel
                (mm)
25 % Offsets
26 geo.offOrigin =[0;0;0];                      % Offset of image from
       origin    (mm)
27 geo.offDetector=[0; 0];                      % Offset of Detector
                (mm)
```

```
28                                              % These two can be also
                                                    defined
29                                              % per angle
30
31 % Auxiliary
32 geo.accuracy=0.5;                            % Variable to define
       accuracy of
33                                              % 'interpolated'
                                                    projection
34                                              % It defines the amoutn
                                                    of
35                                              % samples per voxel.
36                                              % Recommended <=0.5
                                                                (vx/sample
                                                    )
37
38 % Optional Parameters
39 % There is no need to define these unless you actually need them in
       your
40 % reconstruction
41
42
43 geo.COR=0;                                   % y direction
       displacement for
44                                              % centre of rotation
45                                              % correction
                                                                    (mm)
46                                              % This can also be
                                                    defined per
47                                              % angle
48
49 geo.rotDetector=[0;0;0];                     % Rotation of the
       detector, by
50                                              % X,Y and Z axis
                                                    respectively. (rad)
51                                              % This can also be
                                                    defined per
52                                              % angle
53
54 geo.mode='cone';                             % Or 'parallel'. Geometry
        type.
55
56
57 %% Update Info components with the elements of the geometry (or
       create Info in case of .HIS)
58 Info.ObjectType = 'Image';
59 Info.NDims = 3;
60 Info.BinaryData = 'True';
61 Info.BinaryDataByteOrderMSB = 'False';
62 Info.CompressedData = 'False';
63 Info.TransformMatrix =  [1 0 0 0 1 0 0 0 1];
```

```
64 Info.Offset = geo.offOrigin';
65 Info.CenterOfRotation = [0 0 0];
66 Info.AnatomicalOrientation = 'RAI';
67 Info.ElementSpacing = geo.dVoxel';
68 Info.DimSize = geo.nVoxel';
69 Info.ElementType = 'MET_FLOAT';
70 Info.ElementDataFile = 'Tigre_FDK_20180726.raw';
71
72 %% Arc (angles, in radiants)
73
74 [angles ucentre vcentre] = extraction();
75 angles = (angles/180)*pi;
76
77 %%
78
79 % ======= RECONSTRUCTION ALGORITHM ======= %
80
81 ReconsMethods = {'FDK', 'OSSART', 'SART', 'CGLS', 'MLEM', 'ASD-POCS'
       };
82 [ReconstructionMethod, OK] = listdlg('ListString', ReconsMethods, '
       SelectionMode', 'multiple', 'PromptString', 'Select the
       reconstruction method to apply :');
83
84 %% Reconstruct Volume - FDK
85
86 if isempty(find(ReconstructionMethod == 1)) == 0 % 1 == FDK
87
88 imgFDK = FDK(Projections,geo,angles,'filter','hann');
89 Info
90
91 %pas n cessaire pour tester l'algorithm de cr er le raw et le mhd
92 directory = uigetdir;
93 filenameMHD = strcat(directory,'\','Tigre_FDK_20180726.mhd');
94 %create MHD file
95 writeMHD(filenameMHD, Info); % Careful with the path!
96
97 filenameRAW = strcat(directory,'\','Tigre_FDK_20180726.raw');
98 fid = fopen(filenameRAW, 'w+'); % Careful with the path!
99 %create RAW
100 fwrite(fid, imgFDK, 'float');
101 fclose(fid);
102
103 end
104
105 %% Reconstruct Volume - OSSART
106
107 if isempty(find(ReconstructionMethod == 2)) == 0 % 2 == OSSART
108
109
110 nIter = str2double(inputdlg('Enter the number of iteration for OSSART
       (ex : 15) :'));
```

```
111
112 imgOSSART = OS_SART(Projections,geo,angles,nIter);
113
114 directory = uigetdir;
115 filenameMHD = strcat(directory,'\','_Tigre_OSSART_20180726_nIter',
    num2str(nIter),'.mhd');
116 writeMHD(filenameMHD, Info); % Careful with the path!
117
118 filenameRAW = strcat(directory,'\','_Tigre_OSSART_20180726_nIter',
    num2str(nIter),'.raw');
119 fid = fopen(filenameRAW, 'w+'); % Careful with the path!
120 fwrite(fid, imgOSSART, 'float');
121 fclose(fid);
122
123 end
124
125 %% Reconstruct Volume - SART
126
127 if isempty(find(ReconstructionMethod == 3)) == 0 % 3 == SART
128
129 nIter = str2double(inputdlg('Enter the number of iteration for SART (
    ex : 15) :'));
130
131 imgSART = SART(Projections,geo,angles,nIter);
132
133 directory = uigetdir;
134 filenameMHD = strcat(directory,'\','_Tigre_SART_20180726_nIter',
    num2str(nIter),'.mhd');
135 writeMHD(filenameMHD, Info); % Careful with the path!
136
137 filenameRAW = strcat(directory,'\','_Tigre_SART_20180726_nIter',
    num2str(nIter),'.raw');
138 fid = fopen(filenameRAW, 'w+'); % Careful with the path!
139 fwrite(fid, imgSART, 'float');
140 fclose(fid);
141
142 end
143
144 %% Reconstruct Volume - CGLS
145
146 if isempty(find(ReconstructionMethod == 4)) == 0 % 4 == CGLS
147
148 nIter = str2double(inputdlg('Enter the number of iteration for CGLS (
    ex : 15) :'));
149
150 [imgCGLS, err_cgls] = CGLS(Projections,geo,angles,nIter);
151
152 directory = uigetdir;
153 filename = strcat(directory,'\','_Tigre_CGLS_20180726_nIter',num2str(
    nIter));
154 writeMHD(strcat(filename, '.mhd'), Info); % Careful with the path!
```

```
155
156 fid = fopen(strcat(filename,'.raw'), 'w+'); % Careful with the path!
157 fwrite(fid, imgCGLS, 'float');
158 fclose(fid);
159
160 end
161
162 %% Reconstruct Volume - MLEM
163
164 if isempty(find(ReconstructionMethod == 5)) == 0 % 5 == MLEM
165
166 nIter = str2double(inputdlg('Enter the number of iteration for MLEM (
      ex : 15) :'));
167
168 imgMLEM = MLEM(Projections,geo,angles,nIter);
169
170 directory = uigetdir;
171 filenameMHD = strcat(directory,'\','_Tigre_MLEM_20180726_nIter',
      num2str(nIter),'.mhd');
172 writeMHD(filenameMHD, Info); % Careful with the path!
173
174 filenameRAW = strcat(directory,'\','_Tigre_MLEM_20180726_nIter',
      num2str(nIter),'.raw');
175 fid = fopen(filenameRAW, 'w+'); % Careful with the path!
176 fwrite(fid, imgMLEM, 'float');
177 fclose(fid);
178
179
180 end
181
182 %% Reconstruct Volume - ASD-POCS
183
184 if isempty(find(ReconstructionMethod == 6)) == 0 % 6 == ASD-POCS
185
186 nIter = str2double(inputdlg('Enter the number of iteration for ASD-
      POCS (ex : 15) :'));
187
188 imgASDPOCS = ASD_POCS(Projections,geo,angles,nIter);
189
190 directory = uigetdir;
191 filenameMHD = strcat(directory,'\','_Tigre_ASDPOCS_20180726_nIter',
      num2str(nIter),'.mhd');
192 writeMHD(filenameMHD, Info); % Careful with the path!
193
194 filenameRAW = strcat(directory,'\','_Tigre_ASDPOCS_20180726_nIter',
      num2str(nIter),'.raw');
195 fid = fopen(filenameRAW, 'w+'); % Careful with the path!
196 fwrite(fid, imgASDPOCS, 'float');
197 fclose(fid);
198
199
```

```
end

%% Display slices of the reconstruction


RowsSubplot = 3; ColumnsSubplot = 4;

if exist('imgFDK', 'var')
    img = imgFDK;
elseif exist('imgOSSART', 'var')
    img = imgOSSART;
elseif exist('imgSART', 'var')
    img = imgSART;
elseif exist('imgCGLS', 'var')
    img = imgCGLS;
elseif exist('imgMLEM', 'var')
    img = imgMLEM;
elseif exist('imgASDPOCS', 'var')
    img = imgASDPOCS;
else
    error('No image was reconstructed! Can not display any image!')
end

figure(42)
for f=1:RowsSubplot*ColumnsSubplot
    subplot(RowsSubplot, ColumnsSubplot, f)
    imshow(img(:,:,1+f*floor(geo.nVoxel/(RowsSubplot*ColumnsSubplot))
        ), [])
end
```

B.3 DICOM reconstructions

B.3.1 Create the projections

```
function [ projections1, projections2, ideal_proj1, ideal_proj2 ] =
    projectionsdcm(geo,angles)
%projectionsdcm receives a geometry and an array of angle to then
    create
%the projections starting from the reconstructions.
%
%The head is cut in half: an upper and a lower half. There are both
    noisy
%and ideal projections that are returned.

addpath('C:\Users\alarnould\Documents\Alexandre\TIGRE-master\MATLAB')
    ;

%% load dicoms

%the reconstructed image consist of 144 different layers
%images
head1 = zeros(512,512,144);
```

```
15 head2 = zeros(512,512,144);
16 %head = zeros(512,512,144);
17
18 for i = 1:144
19     num1 = 2029+i;
20     num2 = 1029+i;
21     filename1 = ['C:\Users\alarnould\Documents\Alexandre\
           ImagXEcoReconstruction\Projections\20180322
           _JoeHead_CBCT_Philips\Brain_Spect_Brain_Spect_20180322\
           LOC_XCT__1\IM-0001-' num2str(num1) '.dcm'];
22     filename2 = ['C:\Users\alarnould\Documents\Alexandre\
           ImagXEcoReconstruction\Projections\20180322
           _JoeHead_CBCT_Philips\Brain_Spect_Brain_Spect_20180322\
           LOC_XCT__1\IM-0001-' num2str(num2) '.dcm'];
23     info_head1 = dicominfo(filename1);
24     info_head2 = dicominfo(filename2);
25     head1(:,:,i) = dicomread(info_head1);
26     head2(:,:,i) = dicomread(info_head2);
27 end
28
29 head1 = im2single(head1);
30 head2 = im2single(head2);
31
32
33 %% create the projections
34 ideal_proj1 = Ax(head1,geo,angles,'interpolated');
35 ideal_proj2 = Ax(head2,geo,angles,'interpolated');
36
37 %% adding noise to create more realistic projections
38
39 projections1 = addCTnoise(ideal_proj1,'Poisson',1e5,'Gaussian',[0
       10]);
40 projections2 = addCTnoise(ideal_proj2,'Poisson',1e5,'Gaussian',[0
       10]);
41
42 end
```

B.3.2 Reconstruction script

```
1 % Image Reconstruction Script for ImagX.Eco (Tigre toolbox)
2
3 % /!\ Each time you restart MATLAB and TIGRE, you must run the
      initTigre
4 % file from the Tigre-master\MATLAB directory.
5
6 clear all;
7 close all;
8
9 addpath('C:\Users\alarnould\Documents\Alexandre\TIGRE-master\MATLAB')
      ;
10
11 %% Create projections
```

```
12
13 %the amount of angles you project on
14 ang = 200;
15 angles = linspace(0, 2*pi, ang);
16
17 %the geometry (here the standard tigre geo)
18 % VARIABLE                                   DESCRIPTION
                        UNITS
19 %
   -------------------------------------------------------------------------

20 geo.DSD = 1536;                              % Distance Source
   Detector      (mm)
21 geo.DSO = 1000;                              % Distance Source Origin
           (mm)
22 % Detector parameters
23 geo.nDetector=[512; 512];                                    %
   number of pixels              (px)
24 geo.dDetector=[0.8; 0.8];                                    %
   size of each pixel            (mm)
25 geo.sDetector=geo.nDetector.*geo.dDetector; % total size of the
   detector    (mm)
26 % Image parameters
27 geo.nVoxel=[512;512;144];                    % number of voxels
                 (vx)
28 geo.sVoxel=[512;512;144];                    % total size of the image
           (mm)
29 geo.dVoxel=geo.sVoxel./geo.nVoxel;          % size of each voxel
                (mm)
30 % Offsets
31 geo.offOrigin =[0;0;0];                      % Offset of image from
   origin    (mm)
32 geo.offDetector=[0; 0];                      % Offset of Detector
                (mm)
33
34
35 % Auxiliary
36 geo.accuracy=0.5;                            % Accuracy of FWD proj
           (vx/sample)
37
38
39 [projections1, projections2, ideal_proj1, ideal_proj2] =
   projectionsdcm(geo,angles);
40
41 %% show the projections
42
43 %% Plot ideal projections
44
45 % figure
46 % imshow(projections(:,:,50),[])
47
```

```
48 % plotProj(ideal_proj1, angles)
49 % plotProj(ideal_proj2, angles)
50 %
51 %% plot noisy projections
52 %
53 % plotProj(projections1,angles)
54 % plotProj(projections2,angles)
55
56 %% Create the reconstructions
57
58 % ======= RECONSTRUCTION ALGORITHM ======= %
59
60 ReconsMethods = {'FDK', 'OSSART', 'SART', 'CGLS', 'MLEM', 'ASD-POCS',
       'SIRT'};
61 [ReconstructionMethod, OK] = listdlg('ListString', ReconsMethods, '
       SelectionMode', 'multiple', 'PromptString', 'Select the
       reconstruction method to apply :');
62 iter = 0;
63 %% Reconstruct Volume - FDK
64
65 if isempty(find(ReconstructionMethod == 1)) == 0 % 1 == FDK
66 tic
67 imgFDK = FDK(ideal_proj1,geo,angles,'filter','hann');
68 toc
69 end
70
71 %% Reconstruct Volume - OSSART
72
73 if isempty(find(ReconstructionMethod == 2)) == 0 % 2 == OSSART
74
75
76 nIter = str2double(inputdlg('Enter the number of iteration for OSSART
       (ex : 15) :'));
77 iter = nIter;
78 tic
79 imgOSSART = OS_SART(projections1,geo,angles,nIter);
80 toc
81 end
82
83 %% Reconstruct Volume - SART
84
85 if isempty(find(ReconstructionMethod == 3)) == 0 % 3 == SART
86
87 nIter = str2double(inputdlg('Enter the number of iteration for SART (
       ex : 15) :'));
88 iter = nIter;
89 tic
90 imgSART = SART(projections1,geo,angles,nIter);
91 toc
92
93 end
```

```
94
95 %% Reconstruct Volume - CGLS
96
97 if isempty(find(ReconstructionMethod == 4)) == 0 % 4 == CGLS
98
99 nIter = str2double(inputdlg('Enter the number of iteration for CGLS (
       ex : 15) :'));
100 iter = nIter;
101 tic
102 [imgCGLS, err_cgls] = CGLS(projections1,geo,angles,nIter);
103 toc
104 end
105
106 %% Reconstruct Volume - MLEM
107
108 if isempty(find(ReconstructionMethod == 5)) == 0 % 5 == MLEM
109
110 nIter = str2double(inputdlg('Enter the number of iteration for MLEM (
       ex : 15) :'));
111 iter = nIter;
112 tic
113 imgMLEM = MLEM(projections1,geo,angles,nIter);
114 toc
115 end
116
117 %% Reconstruct Volume - ASD-POCS
118
119 if isempty(find(ReconstructionMethod == 6)) == 0 % 6 == ASD-POCS
120
121 nIter = str2double(inputdlg('Enter the number of iteration for ASD-
       POCS (ex : 15) :'));
122 iter = nIter;
123 tic
124 imgASDPOCS = ASD_POCS(projections1,geo,angles,nIter);
125 toc
126 end
127 %% Reconstruct Volume - SIRT
128
129 if isempty(find(ReconstructionMethod == 7)) == 0 % 7 == SIRT
130
131 nIter = str2double(inputdlg('Enter the number of iteration for SIRT (
       ex : 15) :'));
132 iter = nIter;
133 tic
134 imgSIRT = SIRT(ideal_proj1,geo,angles,nIter);
135 toc
136 end
137
138 %% Display slices of the reconstruction
139
140
```

```
str = '';
if exist('imgFDK', 'var')
    img = imgFDK;
    str = ['FDK_low_id_ang' num2str(ang)];
elseif exist('imgOSSART', 'var')
    img = imgOSSART;
    str = ['OSSART_low_noi_ang' num2str(ang) '_iter' num2str(iter)];
elseif exist('imgSART', 'var')
    img = imgSART;
    str = ['SART_low_noi_ang' num2str(ang) '_iter' num2str(iter)];
elseif exist('imgCGLS', 'var')
    img = imgCGLS;
    str = ['CGLS_low_noi_ang' num2str(ang) '_iter' num2str(iter)];
elseif exist('imgMLEM', 'var')
    img = imgMLEM;
    str = ['MLEM_low_noi_ang' num2str(ang) '_iter' num2str(iter)];
elseif exist('imgASDPOCS', 'var')
    img = imgASDPOCS;
    str = ['ASDPOCS_low_noi_ang' num2str(ang) '_iter' num2str(iter)];
elseif exist('imgSIRT', 'var')
    img = imgSIRT;
    str = ['SIRT_id_noi_ang' num2str(ang) '_iter' num2str(iter)];
elseif exist('imgSARTTV', 'var')
    img = imgSARTTV;
    str = ['SARTTV_low_id_ang' num2str(ang) '_iter' num2str(iter)];
else
    error('No image was reconstructed! Can not display any image!')
end

filename = ['C:\Users\alarnould\Documents\Alexandre\
   ImagXEcoReconstruction\August01\' str '.mat'];
save(filename,'img');
```

B.3.3 Scripts for the testing

The script below is used for plotting the slices of the reconstructions, computing the metrics and plotting the graphs of the metrics.

```
%clear all;
close all;
%script to compute the image quality matrix.
addpath('C:\Users\alexa\Documents\EPL\Memoire\Code\Testbench\
   Quality_measures');

%% Loading the 3D images

%loading the reference image (DICOM)
ref1 = zeros(512,512,144);

s = 0;
inter = 0;

for i = 1:144
```

```
15      num1 = 2029+i;
16      filename1 = ['C:\Users\alexa\Documents\EPL\Memoire\Code\Testbench
            \LOC_XCT__1\IM-0001-' num2str(num1) '.dcm'];
17      info_head1 = dicominfo(filename1);
18      s = info_head1.RescaleSlope;
19      inter = info_head1.RescaleIntercept;
20      ref1(:,:,i) = (dicomread(info_head1)*s);
21      ref1(:,:,i) = ref1(:,:,i)+inter;
22  end
23
24  file = 'C:\Users\alexa\Documents\EPL\Memoire\Code\Testbench\August01\
        ';
25  name = '_low_noi_ang100';
26
27  FDK = importdata([file 'FDK' name '.mat'])*s+inter;
28  OSSART5 = importdata([file 'OSSART' name '_iter5.mat'])*s+inter;
29  OSSART10 = importdata([file 'OSSART' name '_iter10.mat'])*s+inter;
30  OSSART20 = importdata([file 'OSSART' name '_iter20.mat'])*s+inter;
31  OSSART30 = importdata([file 'OSSART' name '_iter30.mat'])*s+inter;
32  OSSART40 = importdata([file 'SART' name '_iter5.mat'])*s+inter;
33  SART5 = importdata([file 'SART' name '_iter5.mat'])*s+inter;
34  SART10 = importdata([file 'SART' name '_iter10.mat'])*s+inter;
35  SART20 = importdata([file 'SART' name '_iter20.mat'])*s+inter;
36  SART30 = importdata([file 'SART' name '_iter30.mat'])*s+inter;
37  SART40 = importdata([file 'SART' name '_iter40.mat'])*s+inter;
38  SIRT5 = importdata([file 'SIRT' name '_iter5.mat'])*s+inter;
39  SIRT10 = importdata([file 'SIRT' name '_iter10.mat'])*s+inter;
40  SIRT20 = importdata([file 'SIRT' name '_iter20.mat'])*s+inter;
41  SIRT30 = importdata([file 'SIRT' name '_iter30.mat'])*s+inter;
42  SIRT40 = importdata([file 'SIRT' name '_iter40.mat'])*s+inter;
43  MLEM5 = importdata([file 'MLEM' name '_iter5.mat'])*s+inter;
44  MLEM10 = importdata([file 'MLEM' name '_iter10.mat'])*s+inter;
45  MLEM20 = importdata([file 'MLEM' name '_iter20.mat'])*s+inter;
46  MLEM30 = importdata([file 'MLEM' name '_iter30.mat'])*s+inter;
47  MLEM40 = importdata([file 'MLEM' name '_iter40.mat'])*s+inter;
48  CGLS5 = importdata([file 'CGLS' name '_iter5.mat'])*s+inter;
49  CGLS10 = importdata([file 'CGLS' name '_iter10.mat'])*s+inter;
50  CGLS17 = importdata([file 'CGLS' name '_iter17.mat'])*s+inter;
51  % CGLS20 = importdata([file 'CGLS' name '_iter20.mat'])*s+inter;
52  % CGLS30 = importdata([file 'CGLS' name '_iter30.mat'])*s+inter;
53  % CGLS40 = importdata([file 'CGLS' name '_iter40.mat'])*s+inter;
54  ASDPOCS5 = importdata([file 'ASDPOCS' name '_iter5.mat'])*s+inter;
55  ASDPOCS10 = importdata([file 'ASDPOCS' name '_iter10.mat'])*s+inter;
56  ASDPOCS20 = importdata([file 'ASDPOCS' name '_iter20.mat'])*s+inter;
57  ASDPOCS30 = importdata([file 'ASDPOCS' name '_iter30.mat'])*s+inter;
58  ASDPOCS40 = importdata([file 'ASDPOCS' name '_iter40.mat'])*s+inter;
59
60  %% display slice of reconstruction
61
62  figure
63  imshow(ref1(:,:,144),[])
```

```
64 figure
65 imshow(FDK(:,:,144),[])
66 figure
67 imshow(SART5(:,:,144), [])
68 figure
69 imshow(OSSART5(:,:,144), [])
70 figure
71 imshow(SIRT5(:,:,144), [])
72 figure
73 imshow(MLEM5(:,:,144), [])
74 figure
75 imshow(CGLS5(:,:,144), [])
76 figure
77 imshow(ASDPOCS5(:,:,144), [])
78
79
80 %% Compute the metrics
81
82 UQI0  = UQI(ref1, FDK);
83 UQI1 = [UQI(ref1, SART5) UQI(ref1, SART10) UQI(ref1, SART20) UQI(ref1
       , SART30) UQI(ref1, SART40)];
84 UQI2 = [UQI(ref1, OSSART5) UQI(ref1, OSSART10) UQI(ref1, OSSART20)
      UQI(ref1, OSSART30) UQI(ref1, OSSART40)];
85 %UQI3 = [UQI(ref1, CGLS5) UQI(ref1, CGLS10) UQI(ref1, CGLS20) UQI(
      ref1, CGLS30) UQI(ref1, CGLS40)];
86 UQI3 = [UQI(ref1, CGLS5) UQI(ref1, CGLS10) UQI(ref1, CGLS17) ];
87 UQI4 = [UQI(ref1, MLEM5) UQI(ref1, MLEM10) UQI(ref1, MLEM20) UQI(ref1
       , MLEM30) UQI(ref1, MLEM40)];
88 UQI5 = [UQI(ref1, ASDPOCS5) UQI(ref1, ASDPOCS10) UQI(ref1, ASDPOCS20)
       UQI(ref1, ASDPOCS30) UQI(ref1, ASDPOCS40)];
89 UQI6 = [UQI(ref1, SIRT5) UQI(ref1, SIRT10) UQI(ref1, SIRT20) UQI(ref1
       , SIRT30) UQI(ref1, SIRT40)];
90
91 MSSIM0  = MSSIM(ref1, FDK);
92 MSSIM1 = [MSSIM(ref1, SART5) MSSIM(ref1, SART10) MSSIM(ref1, SART20)
      MSSIM(ref1, SART30) MSSIM(ref1, SART40)];
93 MSSIM2 = [MSSIM(ref1, OSSART5) MSSIM(ref1, OSSART10) MSSIM(ref1,
      OSSART20) MSSIM(ref1, OSSART30) MSSIM(ref1, OSSART40)];
94 %MSSIM3 = [MSSIM(ref1, CGLS5) MSSIM(ref1, CGLS10) MSSIM(ref1, CGLS20)
       MSSIM(ref1, CGLS30) MSSIM(ref1, CGLS40)];
95 MSSIM3 = [MSSIM(ref1, CGLS5) MSSIM(ref1, CGLS10) MSSIM(ref1, CGLS17)
      ];
96 MSSIM4 = [MSSIM(ref1, MLEM5) MSSIM(ref1, MLEM10) MSSIM(ref1, MLEM20)
      MSSIM(ref1, MLEM30) MSSIM(ref1, MLEM40)];
97 MSSIM5 = [MSSIM(ref1, ASDPOCS5) MSSIM(ref1, ASDPOCS10) MSSIM(ref1,
      ASDPOCS20) MSSIM(ref1, ASDPOCS30) MSSIM(ref1, ASDPOCS40)];
98 MSSIM6 = [MSSIM(ref1, SIRT5) MSSIM(ref1, SIRT10) MSSIM(ref1, SIRT20)
      MSSIM(ref1, SIRT30) MSSIM(ref1, SIRT40)];
99
100 CC0  = CC(ref1, FDK);
```

```
101 CC1 = [CC(ref1, SART5) CC(ref1, SART10) CC(ref1, SART20) CC(ref1,
        SART30) CC(ref1, SART40)];
102 CC2 = [CC(ref1, OSSART5) CC(ref1, OSSART10) CC(ref1, OSSART20) CC(
        ref1, OSSART30) CC(ref1, OSSART40)];
103 %CC3 = [CC(ref1, CGLS5) CC(ref1, CGLS10) CC(ref1, CGLS20) CC(ref1,
        CGLS30) CC(ref1, CGLS40)];
104 CC3 = [CC(ref1, CGLS5) CC(ref1, CGLS10) CC(ref1, CGLS17) ];
105 CC4 = [CC(ref1, MLEM5) CC(ref1, MLEM10) CC(ref1, MLEM20) CC(ref1,
        MLEM30) CC(ref1, MLEM40)];
106 CC5 = [CC(ref1, ASDPOCS5) CC(ref1, ASDPOCS10) CC(ref1, ASDPOCS20) CC(
        ref1, ASDPOCS30) CC(ref1, ASDPOCS40)];
107 CC6 = [CC(ref1, SIRT5) CC(ref1, SIRT10) CC(ref1, SIRT20) CC(ref1,
        SIRT30) CC(ref1, SIRT40)];
108
109 RMSE0  = RMSE(ref1, FDK);
110 RMSE1 = [RMSE(ref1, SART5) RMSE(ref1, SART10) RMSE(ref1, SART20) RMSE
        (ref1, SART30) RMSE(ref1, SART40)];
111 RMSE2 = [RMSE(ref1, OSSART5) RMSE(ref1, OSSART10) RMSE(ref1, OSSART20
        ) RMSE(ref1, OSSART30) RMSE(ref1, OSSART40)];
112 %RMSE3 = [RMSE(ref1, CGLS5) RMSE(ref1, CGLS10) RMSE(ref1, CGLS20)
        RMSE(ref1, CGLS30) RMSE(ref1, CGLS40)];
113 RMSE3 = [RMSE(ref1, CGLS5) RMSE(ref1, CGLS10) RMSE(ref1, CGLS17) ];
114 RMSE4 = [RMSE(ref1, MLEM5) RMSE(ref1, MLEM10) RMSE(ref1, MLEM20) RMSE
        (ref1, MLEM30) RMSE(ref1, MLEM40)];
115 RMSE5 = [RMSE(ref1, ASDPOCS5) RMSE(ref1, ASDPOCS10) RMSE(ref1,
        ASDPOCS20) RMSE(ref1, ASDPOCS30) RMSE(ref1, ASDPOCS40)];
116 RMSE6 = [RMSE(ref1, SIRT5) RMSE(ref1, SIRT10) RMSE(ref1, SIRT20) RMSE
        (ref1, SIRT30) RMSE(ref1, SIRT40)];
117
118 %% Plot the metrics
119
120 x = [5 10 20 30 40];
121 u = [5 10 17];
122
123 figure
124 plot(x,MSSIM1,'-xk','LineWidth',2);hold on;
125 plot(x,MSSIM2,':xm','LineWidth',2);hold on;
126 plot(u,MSSIM3,'-+b','LineWidth',2);hold on;
127 plot(x,MSSIM4,'-*c','LineWidth',2);hold on;
128 plot(x,MSSIM5,':og','LineWidth',2);hold on;
129 plot(x,MSSIM6,'-+r','LineWidth',2);hold on;
130 plot(0,MSSIM0,'o','LineWidth',4, 'MarkerFaceColor', [0,0.7,0])
131 legend({'SART', 'OSSART', 'CGLS', 'MLEM','ASDPOCS', 'SIRT', 'FDK'},'
        FontSize',14,'Location','best')
132 xlabel('Iterations', 'FontSize',14)
133 ylabel('SSIM','FontSize',14)
134
135 %% plot time
136
137 time_id1 = [249.214977 499.010168 957.989280 1398.529382
        1844.389708];
```

```
138 time_id2 = [43.534196 86.834759 172.790709 256.997488 344.974777];
139 time_id3 = [222.866608 444.554933 889.581745 1099.403148
      1114.336768];
140 time_id4 = [38.529012 74.991090 147.389171 219.826052 292.535753];
141 time_id5 = [318.150238 579.499114 1051.201815 1516.487186
      1996.886449];
142 time_id6 = [35.500420 70.727525 141.374689 212.398740 282.352987];
143 time_id0 = 4.666215;
144
145 time_noi1 = [268.449711 544.834392 1078.783756 1612.820365
      2161.054654];
146 time_noi2 = [44.171410 86.485412 173.142611 260.043125 347.224544];
147 time_noi3 = [223.925140 445.495054 752.035802];
148 time_noi4 = [39.158704 76.000492 149.507544 223.375857 297.049795];
149 time_noi5 = [321.438376 575.126702 1049.533527 1530.701046
      2004.722641];
150 time_noi6 = [35.432123 70.604573 140.859170 211.770985 282.100895];
151 time_noi0 = 4.760616;
152
153 figure
154 plot(x,time_noi1,'-xk','LineWidth',2);hold on;
155 plot(x,time_noi2,':xm','LineWidth',2);hold on;
156 plot(u,time_noi3,'-+b','LineWidth',2);hold on;
157 plot(x,time_noi4,'-*c','LineWidth',2);hold on;
158 plot(x,time_noi5,':og','LineWidth',2);hold on;
159 plot(x,time_noi6,'-+r','LineWidth',2);hold on;
160 plot(0,time_noi0,'o','LineWidth',4, 'MarkerFaceColor', [0,0.7,0])
161 legend({'SART', 'OSSART', 'CGLS', 'MLEM','ASDPOCS', 'SIRT', 'FDK'},'
      FontSize',14,'Location','northwest')
162 xlabel('Iterations', 'FontSize',14)
163 ylabel('Total time[s]','FontSize',14)
```

Whole bibliography

[1] Guillaume Janssens John Lee and Edmond Sterpin. *Engineering challenges in protontherapy.* (Available at https://moodleucl.uclouvain.be/course/view.php?id=11642). 2017 – 2018.

[2] Frank Peeters. *Chapter 3: Image reconstruction from projections.* (Available at https://moodleucl.uclouvain.
2017 – 2018.

[3] A. H. Andersen and A. C. Kak. "Simultaneous Algebraic Reconstruction Technique (SART): a superior implementation of the ART algorithm". In: *Ultrasonic Imgaging* 6 (1984), pp. 81–94.

[4] L. C. Davis L. A. Feldkamp and J. W. Kress. "Practical cone-beam algorithm". In: *Optical Society of America* 1.6 (1984), pp. 612–619.

[5] Zhou Wang and Alan C. Bovik. "A Universal Image Quality Index". In: *IEEE SIGNAL PROCESSING LETTERS* XX.Y (2002), pp. 1–4.

[6] Emil Y Sidky and Xiaochuan Pan. "Image Reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization". In: *Physics in Medicine and Biology* 53 (2008), pp. 4777–4807.

[7] Alan Conrad Bovik Mehul Samat Zhou Wang Shalini Gupta and Mia Markey. "Complex Wavelet Structural Similarity: A New Image Similarity Index". In: *IEEE TRANSACTIONS ON IMAGE PROCESSING* 18.11 (2009), pp. 2385–2400.

[8] Zhu L. Suh T. Boyd S. Choi K. Wang J. and Xing L. "Compressed sensing based cone-beam computed tomography reconstruction with a first-order method". In: *Medical Physics* 37 (2010), pp. 5113–5125.

[9] Nick Rawluk. *HIS Image File Format Reader.* 2010. URL: `https://nl.mathworks.com/matlabcentral/fileexchange/26993-his-image-file-format-reader` (visited on 04/17/2018).

[10] F Edward Boas and Dominik Fleischmann. "CT artifacts: Causes and reduction techniques". In: *Imaging in Medicine* 4.2 (2012), pp. 229–240.

[11] D. Titley-Peloquin W. Qiu and M.Soleimani. "Blockwise conjugate gradient methods for image reconstruction in volumetric CT". In: *Computer methods and programs in biomedicine* 108 (2012), pp. 669–678.

[12] Anne Michels. *Comparaison d'algorithmes de reconstruction d'images CBCT.* 2015.

[13] Steven Hancock Ander Biguri Manjit Dosanjh and Manuchehr Soleimani. "TIGRE: A MATLAB-GPU toolbox for CBCT image reconstruction". In: *Biomedical Physics & Engineering Express* 2.5 (2016), pp. 1–4.

[14] F. Ben Bouallègue and D. Mariano-Goulart. "Imagerie médicale. Les fondamentaux: radioanatomie, biophysique, techniques et séméologie en radiologie et médecine nucléaire". In: Elsevier Masson, 2017. Chap. 9 Reconstruction tomographique.

[15] Maria Cohut. *The state of cancer: Are we close to a cure?* 2018. URL: `https://www.medicalnewstoday.com/articles/321106.php` (visited on 07/09/2018).

[16] Sandown Dental. *CBCT scanning*. 2018. URL: http://www.sandowndental.com/cbct-scanning-belfast.html (visited on 07/28/2018).

[17] DICOM. *Overview*. 2018. URL: https://www.dicomstandard.org/about/ (visited on 07/09/2018).

[18] IBA. *Proteus One*. 2018. URL: https://iba-worldwide.com/proton-therapy/proton-therapy-solutions/proteus-one#clinically-optimized-environment (visited on 07/09/2018).

[19] Encyclopedia of Mathematics. *Convex subset*. 2018. URL: http://www.encyclopediaofmath.org/index.php?title=Convex_subset&oldid=33689 (visited on 08/07/2018).

[20] MATLAB. *Corr*. 2018. URL: https://nl.mathworks.com/help/stats/corr.html (visited on 07/15/2018).

[21] Stuart Price. *Coherent Scattering*. 2018. URL: https://radiopaedia.org/articles/coherent-scattering (visited on 07/09/2018).

[22] Stuart Price and Ayush Goel. *Compton Scattering*. 2018. URL: https://radiopaedia.org/articles/compton-effect (visited on 07/09/2018).

[23] Hannah Ritchie and Max Roser. *Cancer*. 2018. URL: https://ourworldindata.org/cancer (visited on 07/09/2018).

[24] Hannah Ritchie and Max Roser. *Causes of death*. 2018. URL: https://ourworldindata.org/causes-of-death (visited on 07/09/2018).

[25] Cancer Research UK. *Cancer survival statistics*. 2018. URL: https://www.cancerresearchuk.org/health-professional/cancer-statistics/survival#heading-Zero (visited on 07/28/2018).

[26] Wikipedia. *Cone beam computed tomography*. 2018. URL: https://en.wikipedia.org/wiki/Cone_beam_computed_tomography (visited on 07/09/2018).

[27] Wikipedia. *Pearson correlation coefficient*. 2018. URL: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient (visited on 07/15/2018).

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**