



UNIVERSITEIT • STELLENBOSCH • UNIVERSITY

Fast generation of digitally reconstructed radiographs for use in
2D-3D image registration

by

Jacobus Everhardus Carstens

*Thesis presented at the University of Stellenbosch in partial
fulfilment of the requirements for the degree of*

Master of Science

Department of Applied Mathematics
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leader: Dr N. Muller

December 2008

Copyright © 2008 University of Stellenbosch
All rights reserved.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature:

J.E. Carstens

Date:

Abstract

Fast generation of digitally reconstructed radiographs for use in 2D-3D image registration

J.E. Carstens

Department of Applied Mathematics

University of Stellenbosch

Private Bag X1, 7602 Matieland, South Africa

Thesis: M.Sc. (Applied Mathematics)

December 2008

Radiotherapy requires that patients be positioned accurately for treatment. An intensity based 2D-3D Image Registration (IR) process can be used as a final step in the positioning of the patient. An X-ray of the patient, called a portal radiograph (PR), is acquired and compared to numerous Digitally Reconstructed Radiographs (DRRs). DRRs are generated from data obtained during a CT scan. The DRRs generated by the IR process represent multiple orientations of the CT data relative to the X-ray imaging system used to acquire the PR. The IR process searches for a rigid transform that produces a DRR most similar to the PR. The optimal transformation is used to align the CT coordinate system with that of the operating room and to quantify the positioning error. The time required for IR is dependent on

1. the time it takes for one cost function evaluation, which is dependent on the time it takes to calculate a DRR and the time it takes to measure how well the DRR and PR match, and
2. the number of objective function evaluations required by the optimisation algorithm.

Van der Bijl (2006) showed that the number of cost function evaluations (and therefore DRR calculations) required can be a few hundred. This study aims to reduce the time required to generate DRRs in order to speed up the image registration process. A reduction in registration time reduces the chances of the patient moving before registration completes and also reduces any discomfort that might be experienced.

A novel implementation exploiting modern hardware is explored and found to be a significant improvement over current methods used. A 50 times performance increase in the computation time of DRRs is achieved over the conventional ray casting approach and image registration is performed in under a minute.

Uittreksel

Vinnige berekening van digitaal herkonstrueerde X-straal beelde vir gebruik in 2D-3D beeld registrasie

(“Fast generation of digitally reconstructed radiographs for use in 2D-3D image registration”)

J.E. Carstens

*Departement Toegepaste Wiskunde
Universiteit van Stellenbosch*

Privaatsak X1, 7602 Matieland, Swid Afrika

Tesis: M.Sc. (Toegepaste Wiskunde)

Desember 2008

Radioterapie vereis dat pasiënte akkuraat geposisioneer word vir behandeling. 'n Intensiteitgebaseerde 2D-3D beeld registrasie (IR) proses kan as 'n finale stap in die pasiënt se posisionering gebruik word. 'n X-straal van die pasiënt, wat 'n Portaal X-straalbeeld (PR) genoem word, word geneem en met verskeie digitaal gerekonstrueerde X-straalbeelde (DRRs) vergelyk. DRRs word bereken van data wat tydens 'n CT skandering ingesamel is. Die DRRs wat deur die IR proses gegenereer word verteenwoordig 'n verskeidenheid oriëntasies van die CT data relatief tot die X-straal stelsel wat gebruik word om die PR te neem. Die IR proses soek vir 'n transformasie wat 'n DRR produseer wat die meeste ooreenstem met die PR. Die optimale transformasie word gebruik om die CT koördinaatstelsel aan die koördinaatstelsel van die behandelingskamer te koppel, sowel as om die posisioneringsfout te kwantifiseer. Die tyd wat deur die IR proses benodig word is afhanklik van

1. die tyd wat benodig word vir 'n enkele kostefunksie evaluasie, wat weer afhanklik is van die tyd wat dit neem om 'n DRR te bereken en die tyd om te meet hoe goed die DRR en die PR ooreenstem, en
2. die aantal kostefunksie evaluasies wat deur die optimeringsalgoritme benodig word.

Van der Bijl (2006) het gewys dat die aantal kostefunksie evaluasies (en daarom DRR berekenings) wat benodig word 'n paar honderd kan wees. Die

doel van hierdie studie is om die tyd wat benodig word om 'n DRR te genereer te verminder en sodoende die beeldregistrasie proses vinniger te maak. 'n Vermindering in registrasietyd verminder die kans dat die pasiënt uit sy posisie beweeg voordat die registrasie voltooi en verminder enige ongerief wat ervaar mag word.

'n Nuwe implementasie wat gebruik maak van moderne hardeware is ondersoek en daar is gevind dat dit 'n beduidende verbetering oor die huidige metodes bied. 'n 50 maal verbetering op die konvensionele benadering is behaal in die berekening van DRRs en die beeldregistrasie geskied in minder as 'n minuut.

Acknowledgements

I would like to thank the following people:

My academic supervisor, Dr. Neil Muller from the University of Stellenbosch, for his guidance during the research.

Evan de Kock from iThemba LABS, for countless ideas and critical assessments.

My wife, Alison, for her support and encouragement throughout this endeavour.

My son, Nicholas, for forcing me to stay up at night.

All my family and friends for their support.

Contents

Declaration	ii
Abstract	iii
Uittreksel	v
Acknowledgements	vii
Contents	viii
List of Figures	xi
List of Tables	xiii
Abbreviations and acronyms	xv
Notation conventions	xvi
1 Introduction	1
1.1 The current patient positioning system	2
1.2 Improving the verification of the patient position	4
1.3 The aim of this thesis	6
1.4 Thesis outline	6
2 Background	7
2.1 CT scans	7
2.2 Digitally Reconstructed Radiographs (DRRs)	8
2.2.1 The X-ray Attenuation Calibration Curve (XACC)	9
2.2.2 Using interpolation	12
2.3 Discussion of the DRR generation literature	12
2.3.1 Ray casting	13
2.3.2 Light fields	18
2.3.3 Shear-warp factorisation	19
2.3.4 Cylindrical harmonics	23
2.3.5 Splatting or voxel projection	24

2.4	Parallelisation and OpenMP	25
2.5	Summary	28
3	Improving DRR generation time	29
3.1	Ray casting: Optimised versions of Siddon's algorithm	29
3.1.1	Optimising Siddon's algorithm through incremental techniques	29
3.1.2	Utilising voxel array symmetry properties	33
3.1.3	Early ray termination	42
3.2	Light fields	43
3.2.1	Using light fields for DRR generation	46
3.2.2	Compression	48
3.3	Parallelisation of the ray cast and light field algorithms	48
3.3.1	Trivial parallelisation	48
3.3.2	Adapting the algorithms	49
3.4	Summary	51
4	Formulating the image registration problem	53
4.1	Quantifying the error	53
4.2	Evaluation of DRR generation methods	54
4.2.1	Qualitative analysis of images	54
4.2.2	Quantitative analysis of images	55
4.3	The optimiser	56
4.3.1	Brent's method	57
4.3.2	Conjugate directions	58
4.3.3	Powell's method	59
4.3.4	The cost function	60
4.4	Formulating the DRR generation problem	61
4.4.1	Coordinate systems	61
4.4.2	Vectors and points used to construct DRRs	62
4.4.3	Transformations	63
4.4.4	Creating a DRR using conventional ray casting	64
4.4.5	Determining the transformations G_{BL} and G_{LB}	65
4.4.6	Constructing the light slab	65
4.4.7	Creating a DRR from a light slab	68
4.5	Summary	70
5	Experiments and results	71
5.1	Similarity tests	72
5.1.1	Similarity measure performance	78
5.1.2	Light slab sampling	78
5.2	Light slab generation time	79
5.3	DRR generation time	82
5.4	Optimisation algorithm performance tests	85

<i>Contents</i>	x
5.4.1 Performance in terms of accuracy	90
5.4.2 Performance in terms of computation time	90
5.4.3 Boundary cases	91
5.5 Summary	91
6 Conclusions and recommendations	93
A System specifications	95
A.1 Hardware	95
A.2 Software	95
B Mathematical aspects	96
B.1 Rotation matrices	96
B.2 Transformations	96
B.3 Determining the intersection of a line with a plane	97
B.4 Mutual Information properties	97
B.5 Steradians	98
B.6 Taylor approximations	98
C Additional optimiser performance results	100
C.1 Focal plane 64×64 , image plane 256×256 , DRR 512×512 and optimum located 0.5mm and 0.5° from $-\epsilon$	100
C.2 Focal plane 32×32 , image plane 512×512 , DRR 512×512 and optimum located at ϵ	103
C.3 Focal plane 64×64 , image plane 256×256 , DRR 512×512 and optimum located at ϵ	106
Bibliography	109

List of Figures

1.1	Proton depth dose curves.	2
1.2	Proton plan images.	3
1.3	An overview of the 2D-3D registration process.	5
2.1	CT slices and the CT cube	8
2.2	A 2D representation of the radiological path	9
2.3	Examples of DRRs	10
2.4	An example of an X-ray Attenuation Calibration Curve	11
2.5	Using an XACC to generate DRRs	11
2.6	The sampling method for normal and interpolated DRRs	12
2.7	Ray intersections	13
2.8	The definition of α_{min} and α_{max}	14
2.9	A ray in a light field coordinate system.	18
2.10	A 2D view of a light slab.	18
2.11	The transformation of a volume to sheared object space for a parallel projection	20
2.12	The transformation of a volume to sheared object space for a perspective projection.	20
2.13	The three steps of shear-warp algorithms.	22
2.14	The projection of a voxel onto a plane.	25
2.15	Parallelisation of an algorithm.	26
3.1	The four types of lines found when observing the geometric symmetry of a ray crossing through pixel space	33
3.2	The different patterns emerging when a ray crosses through pixel space.	34
3.3	Notations used in the 2D description.	34
3.4	Interleaving.	38
3.5	The notation for the 3D ray-tracing algorithm.	38
3.6	Optimising the algorithm.	42
3.7	The 5D representation of a ray	44
3.8	A 2D view of the rays contained in a single light slab.	44

3.9	A 2D view of a combination of four light slabs used to produce a 360° view of an object located in the centre.	45
3.10	A light slab can be interpreted as a 2D collection of 2D images.	46
3.11	Light field plane positions when generating DRRs.	47
3.12	Generating a DRR from a light slab.	47
3.13	An image-order algorithm	49
4.1	An example of a difference image.	55
4.2	A 2D illustration of the unconstrained cost function $c(x)$	61
4.3	The relationship between the various coordinate systems.	63
5.1	Similarity measure for translations along the x axis	74
5.2	Similarity measure for translations along the y axis	75
5.3	Similarity measure for translations along the z axis	75
5.4	Similarity measure for rotations around the x axis	75
5.5	Similarity measure for rotations around the y axis	76
5.6	Similarity measure for rotations around the z axis	76
5.7	Similarity measure for translations along the y and z axes	76
5.8	Similarity measure for rotations around the x and y axes	77
5.9	Similarity measure for translations along the y and z axes and rotations around the x and y axes	77
5.10	Similarity measure for translations and rotations on all the axes	77
5.11	A scaled version of the Correlation Coefficient similarity measure for translations along the x axis.	80
5.12	Similarity measure for translations along the x axis for two other CT orientations.	80
5.13	The perturbed images used in the optimiser tests.	86

List of Tables

5.1	The legend to the similarity test graphs.	74
5.2	The computation times of different light slabs.	81
5.3	Generation time (in seconds) of a 128×128 DRR using the various algorithms	83
5.4	Generation time (in seconds) of a 256×256 DRR using the various algorithms	83
5.5	Generation time (in seconds) of a 512×512 DRR using the various algorithms	84
5.6	Optimiser performance for a perturbation in δ_x	88
5.7	Optimiser performance for a perturbation in δ_y	88
5.8	Optimiser performance for a perturbation in δ_z	88
5.9	Optimiser performance for a perturbation in θ_x	88
5.10	Optimiser performance for a perturbation in θ_y	88
5.11	Optimiser performance for a perturbation in θ_z	89
5.12	Optimiser performance for perturbations in δ_y and δ_z	89
5.13	Optimiser performance for perturbations in θ_x and θ_y	89
5.14	Optimiser performance for perturbations in δ_y , δ_z , θ_x and θ_y	89
5.15	Optimiser performance for perturbations in δ_x , δ_y , δ_z , θ_x , θ_y , θ_z	89
C.1	Optimiser performance for a perturbation in δ_x	100
C.2	Optimiser performance for a perturbation in δ_y	100
C.3	Optimiser performance for a perturbation in δ_z	101
C.4	Optimiser performance for a perturbation in θ_x	101
C.5	Optimiser performance for a perturbation in θ_y	101
C.6	Optimiser performance for a perturbation in θ_z	101
C.7	Optimiser performance for perturbations in δ_y and δ_z	101
C.8	Optimiser performance for perturbations in θ_x and θ_y	102
C.9	Optimiser performance for perturbations in δ_y , δ_z , θ_x and θ_y	102
C.10	Optimiser performance for perturbations in δ_x , δ_y , δ_z , θ_x , θ_y , θ_z	102
C.11	Optimiser performance for a perturbation in δ_x	103
C.12	Optimiser performance for a perturbation in δ_y	103
C.13	Optimiser performance for a perturbation in δ_z	103
C.14	Optimiser performance for a perturbation in θ_x	103

C.15	Optimiser performance for a perturbation in θ_y	104
C.16	Optimiser performance for a perturbation in θ_z	104
C.17	Optimiser performance for perturbations in δ_y and δ_z	104
C.18	Optimiser performance for perturbations in θ_x and θ_y	104
C.19	Optimiser performance for perturbations in δ_y , δ_z , θ_x and θ_y	104
C.20	Optimiser performance for perturbations in δ_x , δ_y , δ_z , θ_x , θ_y , θ_z	105
C.21	Optimiser performance for a perturbation in δ_x	106
C.22	Optimiser performance for a perturbation in δ_y	106
C.23	Optimiser performance for a perturbation in δ_z	106
C.24	Optimiser performance for a perturbation in θ_x	106
C.25	Optimiser performance for a perturbation in θ_y	107
C.26	Optimiser performance for a perturbation in θ_z	107
C.27	Optimiser performance for perturbations in δ_y and δ_z	107
C.28	Optimiser performance for perturbations in θ_x and θ_y	107
C.29	Optimiser performance for perturbations in δ_y , δ_z , θ_x and θ_y	107
C.30	Optimiser performance for perturbations in δ_x , δ_y , δ_z , θ_x , θ_y , θ_z	108

Abbreviations and acronyms

2D	Two-dimensional
3D	Three-dimensional
nD	n-dimensional
CC	Correlation Coefficient
CCC	Cross Correlation Coefficient
CCD	Charge Coupled Device
CPU	Central Processing Unit
CT	Computed Tomography
DRR	Digitally Reconstructed Radiograph
IR	Image Registration
GB	Gigabyte
MI	Mutual Information
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
PET	Positron Emission Tomography
PR	Portal Radiograph
RMSE	Root Mean Squared Error
SMP	Symmetric Multiprocessing
SPG	Stereophotogrammetry
SPMD	Single Program Multiple Data
XACC	X-ray Attenuation Calibration Curve

Notation conventions

s A scalar value.

\mathbf{v} A vector.

\mathcal{B} A coordinate system.

R A matrix.

\mathbf{t} A translation vector.

$G_{AB} = (R_{AB}, \mathbf{t}_{AB})$ A transformation from coordinate system \mathcal{B} to coordinate system \mathcal{A} .

G_E The error transformation.

$\|\mathbf{v}\|_2$ L_2 norm of vector \mathbf{v} .

$|s|$ The absolute value of s .

$\lfloor s \rfloor$ The floor of value s .

$\lceil s \rceil$ The ceiling of value s .

P A point on a line.

SE A line segment from point S to point E .

$\|SE\|$ The length of the line segment SE .

Chapter 1

Introduction

iThemba LABS (Laboratory for Accelerator Based Sciences) is an organisation funded by the National Research Foundation with the mandate to promote basic and applied research using particle beams, particle radiotherapy for the treatment of certain types of lesions and the production of radioactive isotopes for nuclear medicine research.

The medical radiation department provides proton and neutron therapy. Both types of therapy make use of a k-200 separated sector cyclotron. The cyclotron accelerates protons to energies of up to 200MeV and heavier particles to much higher energies. Neutrons and protons both have approximately the same mass. However, because protons are charged particles and neutrons are uncharged, they have very different physical properties and biological effects.

Protons have a physical dose distribution that is useful for certain types of cancer treatment, like stereotactic radiosurgery. Although it is called *radiosurgery*, it does not involve cutting or manipulation of the affected parts of the body. Radiosurgery involves the delivery of one high dose or more smaller doses of radiation beams which focus on the tumour. Stereotactic radiosurgery makes use of immobilisation and positioning devices to accurately position a patient. Accurate positioning enables radiosurgery to be used for tumours or abnormalities that lie close to sensitive organs.

Stereotactic radiosurgery is performed in a single session. When a treatment is spread over a few days or weeks it is referred to stereotactic *radiotherapy*, which is a fractionated treatment.

When stereotactic radiosurgery is performed the DNA of the tumour cells is altered and it loses its ability to reproduce. The tumour usually shrinks over a period of a few months.

The green curve in figure 1.1 shows the dose distribution for protons in water. It can be seen that the relative dose increases steadily and then suddenly peaks and drops to zero. This peak is called the *Bragg peak* and it is the characteristic of protons that makes it very attractive for treating tumours that are located close to sensitive organs like the eyes or spinal chord, where

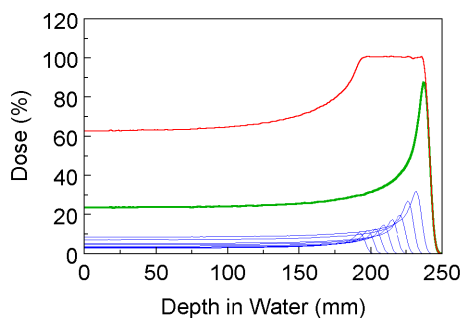


Figure 1.1: Proton depth dose curves. The green curve is the normal dose distribution of protons in water. The blue curves represent modulated beams. The red curve is the sum of the blue curves and the green curve, which is how a spread-out Bragg peak is constructed.

other types of radiation might inflict too much damage to the healthy tissue. By altering the energy, protons can be stopped at any particular point in the body. This means that all organs beyond that range are protected. The peak can also be spread out to efficiently target the whole tumour. The high precision required in proton therapy requires that a patient be positioned very accurately in the proton beam.

Proton therapy at iThemba LABS is provided using a fixed horizontal beam line. The beam characteristics are controlled by various devices. A double scattering system provides a uniform dose. Field-specific collimators are used to shape the beam to match the tumour shape and range modulators vary the dose peak. Together these devices can provide a dose distribution that conforms well to the target volume.

Currently a treatment planning system called VOXELPLAN is used at iThemba LABS. VOXELPLAN uses the VIRTUOS (VIRTUal radiOtherapy Simulator) system developed by the German Cancer Research Centre in Heidelberg, Germany. VIRTUOS is used to view CT data, delineate volumes of interest, define and display treatment beams and to display calculated dose distributions superimposed on CT images. The dose distributions are calculated by a software system developed at iThemba LABS and this system together with VIRTUOS makes up the VOXELPLAN system (de Kock, 2004). Figure 1.2 shows some views of a patient plan.

1.1 The current patient positioning system

In the current system the positioning of a patient is accomplished with the aid of a number of devices.

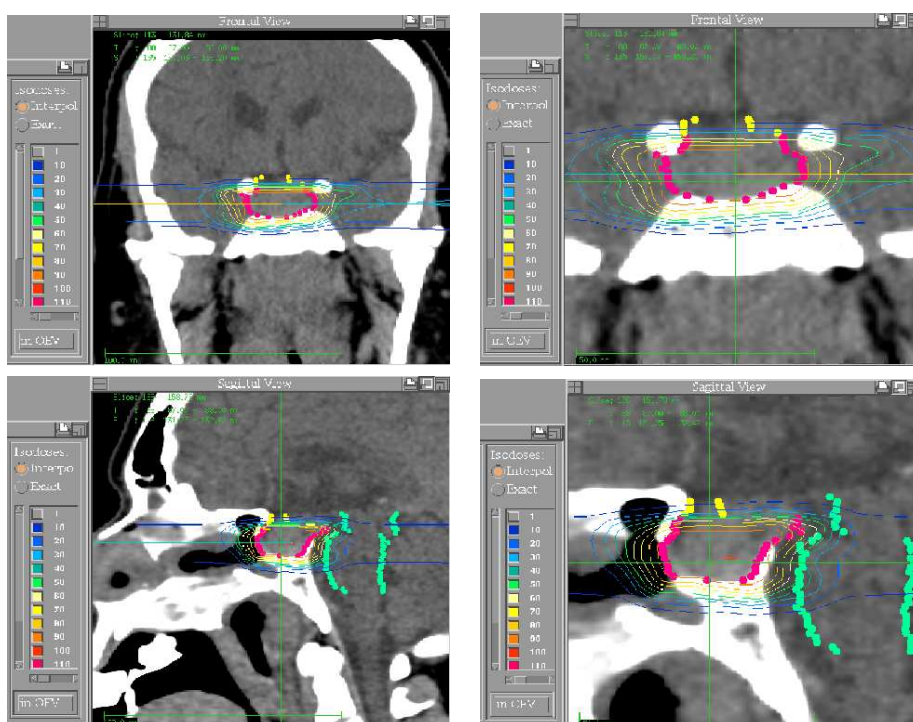


Figure 1.2: A frontal view of a proton plan (top left), an enlargement of the frontal view (top right), the sagittal view of the same plan (bottom left) and an enlargement of the sagittal view (bottom right). The dots in the images are used to delineate the optical chiasma, brain stem (both sensitive organs) as well as the tumour. The isodose lines indicate the amount of dose delivered to an area.

The mask The first device is a close-fitting, patient specific mask with radiopaque and retro-reflective markers located on it. A CT scan of the patient is taken with the mask fitted to the patient. This establishes a relationship between the marker positions and the anatomy of the patient.

The SPG system The second device is a real-time stereophotogrammetry (SPG) system. The SPG system contains a number of charge coupled device (CCD) cameras which detect the markers and are able to compute their respective positions in a 3D coordinate system.

The chair The patient is placed on a motorised chair and an immobilisation device is used to fix the mask and, theoretically, the patient to the chair. The motorised chair is then instructed by the SPG system to move the markers and, by implication, the patient to the position required for treatment. The chair can only perform translations and roll and limited pitch rotations. This, coupled with the fact that only a fixed horizontal beam line is available, restricts the types of lesions that can

be treated (de Kock, 2004; van der Bijl, 2006).

Fitting the mask to the patient on different occasions opens up the possibility of small differences being introduced in the relative positions of the markers and the patient anatomy. When treating a patient, it is necessary to verify that the patient's anatomy is correctly positioned according to the treatment plan before the patient can be treated. This verification is currently accomplished by visual comparison of a portal radiograph (PR) taken when the patient is positioned and a digitally reconstructed radiograph (DRR), generated by the treatment planning system, in which the patient has the correct treatment position. The PR is currently obtained by using an X-ray source upstream of the collimator in the beam line to generate a radiograph of the patient on an X-ray film. The verification procedure described is manual, time-consuming and needs to be repeated for each of the treatment fields (de Kock, 2004).

1.2 Improving the verification of the patient position

The current verification system has two major shortcomings. The first is the fact that the manual nature of the process is labour intensive and time consuming. Sarrut & Clippe (2003) also reported this shortcoming. Prolonged treatment time adds to patient discomfort and increases the risk of the patient moving, which implies another iteration of the positioning procedure. The second major shortcoming is that visual inspection is prone to errors. Because proton therapy is used for treatment of tumours close to sensitive organs, these errors will be detrimental to the patient's health.

Van der Bijl (2006) proposes a solution that verifies the patient position automatically and accurately, thus overcoming the shortcomings of the current system. The proposed system uses a digital X-ray detector to acquire portal radiographs (PRs), as opposed to X-ray film. After a patient has been positioned, the deviation of his position from the correct treatment position can be described by a rigid body transformation G_E , called the error transformation. The size of the positioning error can be quantified from G_E . This quantification can be used to

- verify that the patient is positioned within certain acceptable limits and
- adjust the patient position using G_E if the patient is positioned outside these limits.

The error transformation G_E includes possible positioning errors by the SPG system and misalignment errors introduced when fitting the mask to the patient. The accuracy of the SPG system is high. Recently a new mask fitting system was introduced. A bite block is created for each patient and fitted to the mask. This has proven to be more accurate than the previous

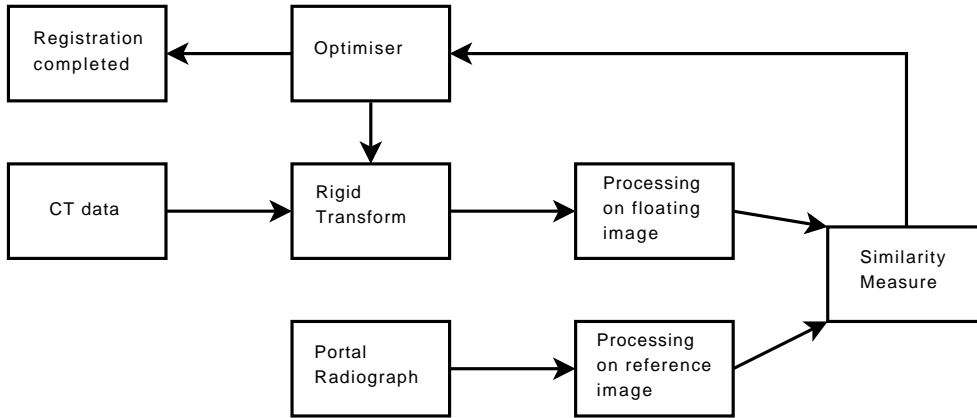


Figure 1.3: An overview of the 2D-3D registration process. A PR (reference image) is compared to various DRRs (floating images) using a similarity measure. The optimiser searches for the rigid transformation that produces a DRR most similar to the PR until registration completes.

methods (de Kock, 2008). One can safely assume that the components of G_E are all smaller than 5mm and 5° for the translations and rotations, respectively.

Image registration is a process whereby the spatial correspondence between two coordinate spaces are established. The result is a transformation linking the two spaces. In this problem we want to establish G_E , the error between the treatment position and the observed position.

2D-3D image registration is a process where 3D CT data acquired pre-operatively are registered to a 2D PR image obtained intra-operatively (Rusakoff *et al.*, 2003). The PR is an X-ray image of the patient's anatomy after being positioned. It is compared to various digitally reconstructed radiographs (DRRs) calculated from the CT data. A DRR is a synthetic image that approximates the physics involved when an X-ray image is generated. Simplistically the DRR pixel values are the summation of attenuation coefficients based on the CT values encountered along a ray casted camera geometry. Section §2.2 gives a more thorough description. A DRR is compared to the PR using a similarity measure. The optimiser thus searches for the transformation that produces a DRR most similar to the PR. A schematic representation of the process is shown in figure 1.3.

Mathematical description The cost function of the image registration optimiser is

$$c(G'_E) = SM(P, D(G'_E)) \quad , \quad (1.2.1)$$

where G'_E is the estimated error transformation chosen for an iteration of the optimisation process, P is the PR image containing the current treatment position, D is a function that returns a DRR and SM is a similarity measure

of how well the two images compare. G_E is calculated by optimising the cost function c .

1.3 The aim of this thesis

Van der Bijl (2006) showed that his proposed solution is accurate, robust and automatic. However, an important outstanding requirement for the system to be usable in practice is that it be fast. The longer a patient needs to wait for the registration process to complete, the higher the probability that the patient will move out of his initial position. The generation of DRRs is computationally expensive and since hundreds of DRRs may be required for the registration process, it is important to speed up DRR generation. Van der Bijl (2006) reported that his image registration implementation took about 7.5 minutes to verify the patient position. He postulated that bringing the DRR generation time down to 0.5 seconds would enable the registration process to be completed in less than a minute.

The aim of this study is to find a fast DRR generation algorithm. In particular we only have to cater for the creation of DRR images that are contained in a known limited vicinity.

Though van der Bijl suggested that a DRR must be generated in under 0.5 seconds for a practical implementation to be viable, for the purposes of this thesis a broader constraint will be applied. The constraint is that the image registration process completes in less than three minutes, which is the minimum time the current manual verification process takes (de Kock, 2008). Furthermore it must be shown that the new DRR generation algorithm does not destroy the accuracy or robustness of the registration process.

1.4 Thesis outline

The remainder of the thesis is laid out as follows: Chapter 2 covers the literature review and other topics relevant to the problem, Chapter 3 details ways in which DRR generation can be sped up, Chapter 4 gives the mathematical formulation of the problem, Chapter 5 provides a presentation and discussion of the experiments and the results and Chapter 6 covers the conclusions and recommendations.

Chapter 2

Background

This chapter provides the background information required to formulate the rest of the thesis. Section §2.1 introduces CT scans, which are used to construct DRRs as described in section §2.2. Section §2.3 presents the various DRR optimisation algorithms proposed in the literature and section §2.4 details the theoretical aspects of parallelisation and provides an example implementation.

2.1 CT scans

A CT scanner is used to conduct a study of the patient and produces a 3D image of the anatomy. The CT data are made up of a set of image slices (Figure 2.1(left)). The slices have a specific thickness and are a constant distance apart from each other, typically 2mm or 4mm. A slice contains pixels with fixed dimensions (typically 1mm × 1mm) and each pixel is assigned a value. A typical CT data set might consist of 50 slices, where each slice contains 512 × 512 pixels. Modern CT scanners are able to produce several hundred slices consisting of 1024 × 1024 pixels (Van der Bijl, 2006). The volume consisting of the collection of slices is referred to as the CT *cube* and is divided into volume elements, called voxels (Figure 2.1(right)). Note that in practice the CT data usually take the shape of a cuboid, but it is referred to as a cube regardless. A voxel represents a volume of the patient’s anatomy and has an associated CT number H , also called the Hounsfield number. In X-ray computed tomography the Hounsfield number is a measure of the X-ray attenuation properties of a material, given by

$$H = 1000 \left(\frac{\bar{\mu}_m - \bar{\mu}_w}{\bar{\mu}_w} \right) \quad , \quad (2.1.1)$$

where $\bar{\mu}_m$ is a measure of the X-ray attenuation of the material inside the voxel, which is dependent on the specific CT scanner used and discussed further in section §2.2.1. $\bar{\mu}_w$ is the average attenuation coefficient of water for the energy spectrum of the CT scanner determined during calibration of the CT

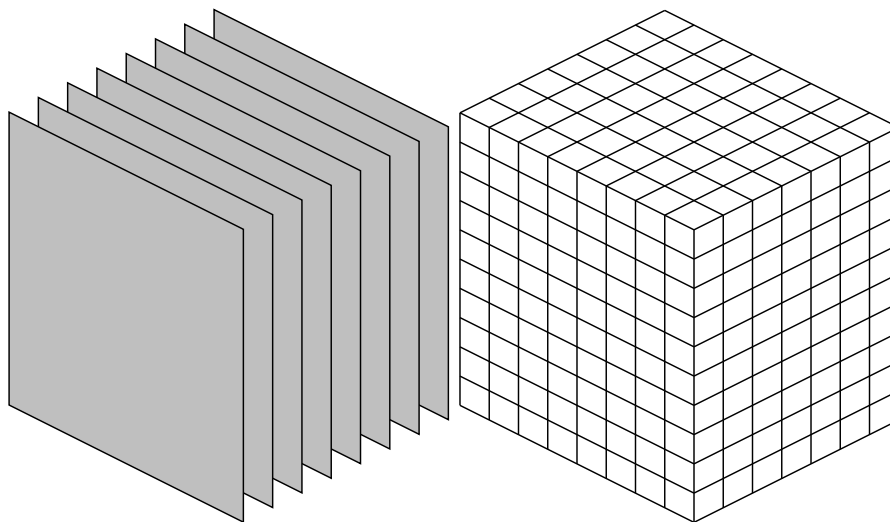


Figure 2.1: CT slices made up of pixels (left) and the CT cube constructed from voxels (right).

scanner with a water phantom. The Hounsfield number usually lies in the range $[-1000..2500]$, with -1000 used to represent air, 0 water, values around 1500 bone and values over 2000 teeth. Since a voxel may contain different types of material, the Hounsfield number represents the average attenuation of the material contained in the voxel.

2.2 Digitally Reconstructed Radiographs (DRRs)

Informally a DRR is a projection of CT data onto a specified plane. In order to create a more formal definition, the concept of a *radiological path length* is introduced. Let $\rho(i, j, k)$ denote the voxel density in a 3-dimensional volume and $l(i, j, k)$ the intersection length of a ray with a voxel, then the radiological path length may be written as

$$d = \sum_i \sum_j \sum_k l(i, j, k) \rho(i, j, k) \quad . \quad (2.2.1)$$

Figure 2.2 shows a 2D representation of the radiological path. Note that $\rho(i, j, k)$ is a function that maps the Hounsfield number of voxel (i, j, k) to a specific attenuation coefficient, which is the XACC described in section §2.2.1. A DRR thus represents a collection of pixel values, where each pixel value is the radiological path length from an X-ray source to its centre. Figure 2.3 shows examples of DRRs.

The radiological path length is proportional to the energy transfer function of X-rays to X-ray film. In all the literature surveyed and for the purposes

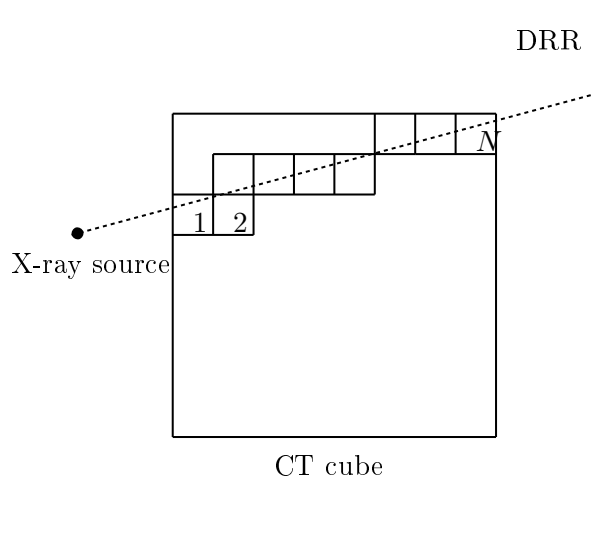


Figure 2.2: A 2D representation of the radiological path of a ray through the CT cube.

of the thesis it is assumed that a digital X-ray detector exhibits the same characteristics as the X-ray film.

The DRRs in figure 2.3 are generated from a CT cube with 58 slices, each with a pixel resolution of 256×256 . The pixel dimensions are 1.016mm and the slices are 4mm apart. The result of the relatively large spacing between slices, which translates to a lower sampling density, can clearly be seen in the first two images where the viewing direction is parallel to the CT slices. The third image is created with a viewing direction orthogonal to the CT slices and it is evident that the higher sampling density leads to a more realistic DRR being generated.

2.2.1 The X-ray Attenuation Calibration Curve (XACC)

The X-ray attenuation calibration curve is a mapping from Hounsfield numbers to attenuation coefficients. The XACC maps the Hounsfield numbers from the CT scanner to the approximate attenuation coefficients observed when using the X-ray system to obtain a PR. Using these attenuation coefficient provides a better approximation to the radiological path length when evaluating equation (2.2.1). The attenuation coefficients observed by the CT and X-ray imaging systems differ because of the varying parameters between the two systems, like the tube voltages used, the anode angle and various filtration parameters. Figure 2.4 shows a typical XACC. Figure 2.5 shows the difference between a DRR computed using an XACC and one computed without using an XACC. Van der Bijl (2006) provides a description with more technical details on the

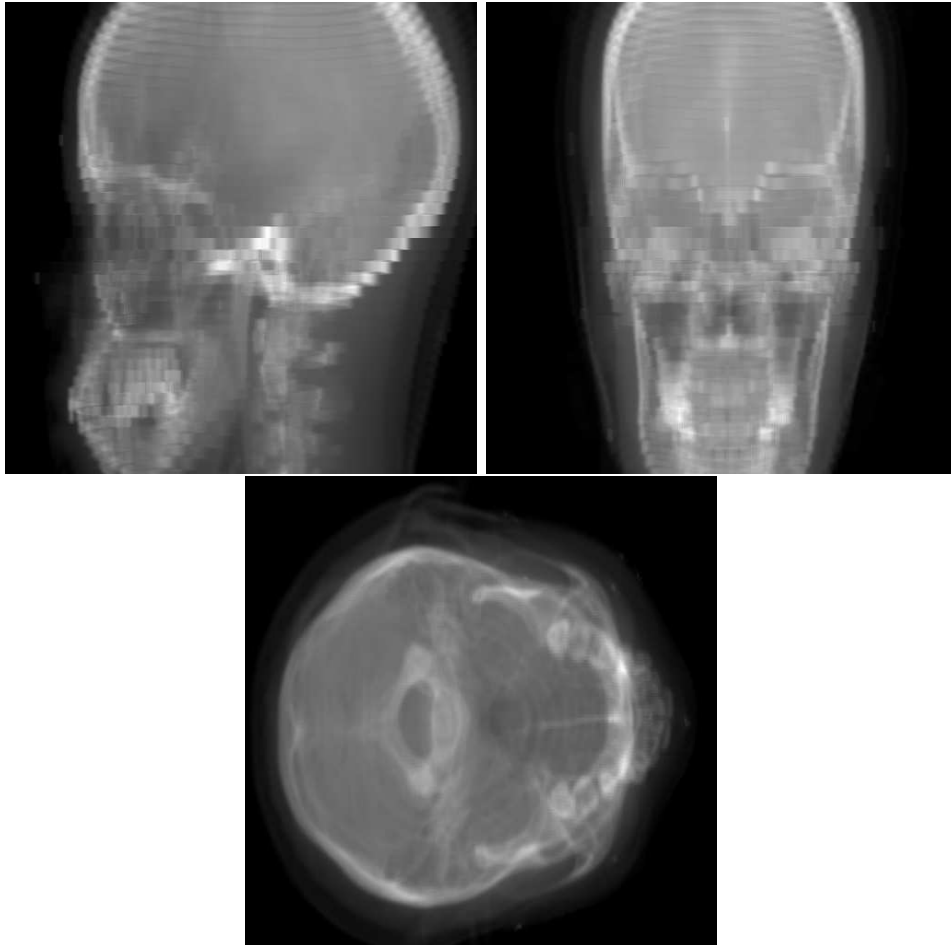


Figure 2.3: Examples of DRRs generated from the same CT data, but from three different angles. The first two images suffer from the fact that the slice distance in the CT scan is quite high. The third image does not suffer from this as it was generated with a viewing direction orthogonal to the slices.

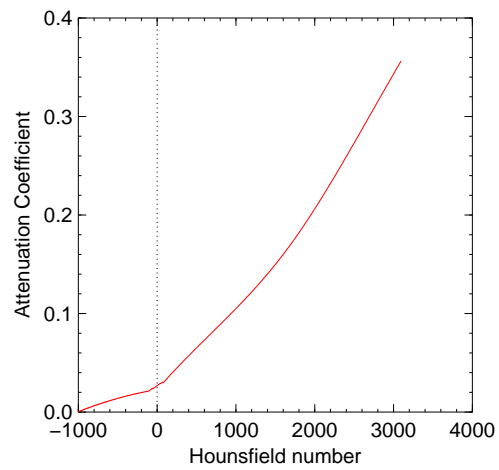


Figure 2.4: An example of an X-ray Attenuation Calibration Curve

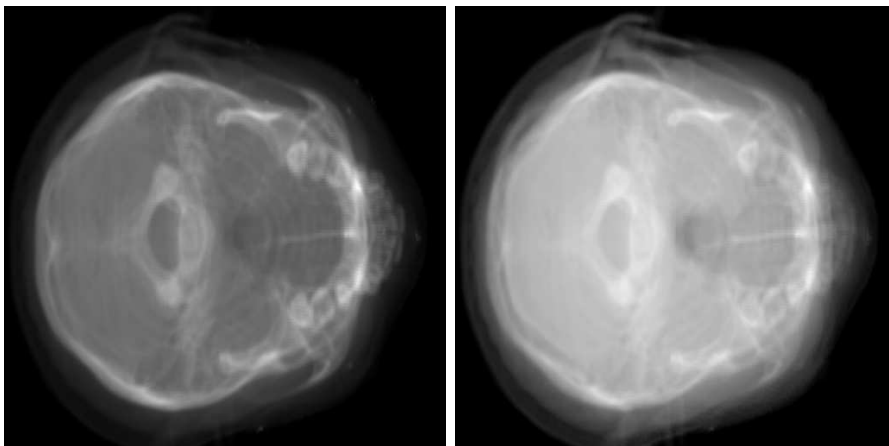


Figure 2.5: An example of a DRR generated using an XACC (left) and a DRR generated without using an XACC (right).

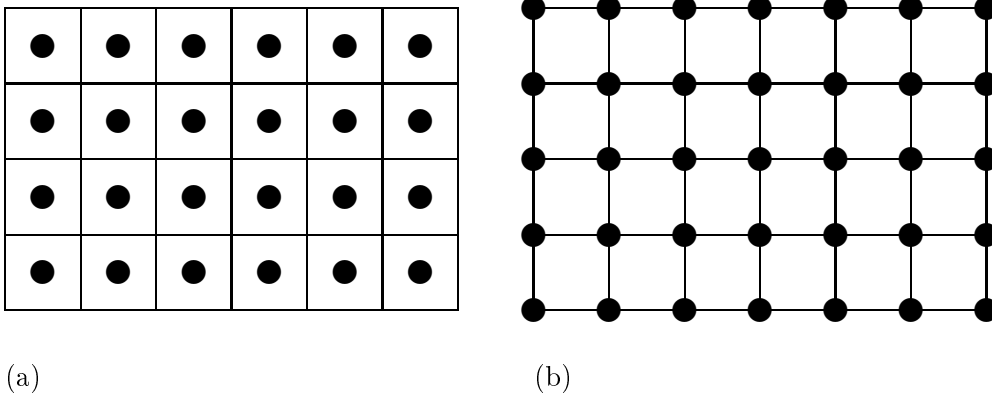


Figure 2.6: The sampling method used for (a) normal DRR generation and (b) interpolated DRR generation. The circles indicate where the sample values are calculated relative to the eventual pixels, which are indicated as squares in the figure.

construction of the XACC.

2.2.2 Using interpolation

Low resolution DRRs suffer from the fact that a single ray value is computed for each pixel in the image. A better approximation, without increasing the sampling resolution significantly, can be achieved using linear interpolation. With interpolation 4 samples are used as input to a bilinear interpolation function. This function then determines the value of the pixel. Figure 2.6 shows the different samples taken when generating a DRR with and without interpolation. Finding the interpolated value at the centre of each pixel is a special case of the bilinear interpolation algorithm. In this case the algorithm reduces to simply averaging the 4 samples:

$$v_{interp} = \frac{1}{4} \sum_{i=0}^3 v_i \quad ,$$

where $\{v_i\}$ are the values of the four corners of the pixel and v_{interp} is the interpolated value.

Bilinear interpolation is preferred over other methods, like cosine interpolation, a cubic fit or a spline, because it is faster to compute.

2.3 Discussion of the DRR generation literature

This section briefly discusses the various DRR generation algorithms investigated. These algorithms can be classified into two types based on what data structure is traversed during construction of a DRR.

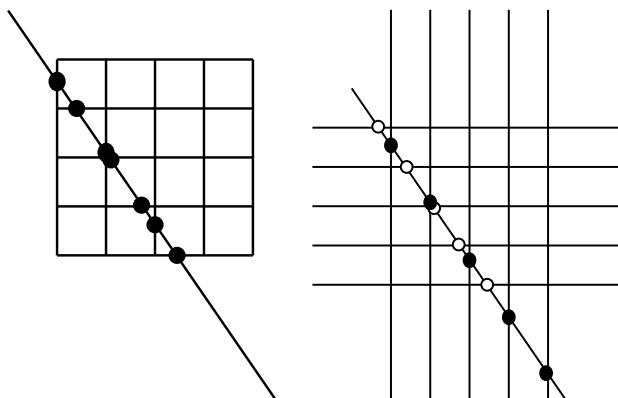


Figure 2.7: A ray intersecting with pixels (left) and a ray intersecting with lines (right).

Image-order algorithms are algorithms that operate by iterating over the resulting image and casting rays to each image pixel, processing the voxels encountered along each ray. The CT data structure must be traversed for every ray. This leads to redundant computations (Lacroute and Levoy, 1994).

Object-order algorithms are algorithms that operate by projecting voxels onto the image while traversing the CT data in storage order (Lacroute and Levoy, 1994).

2.3.1 Ray casting

When evaluating equation (2.2.1) directly, the algorithm scales with the number of voxels in the CT array. Siddon (1985) proposes a new method that scales with the linear dimensions of the CT array.

In his paper, Siddon proposes viewing voxels as the intersection of equally spaced, parallel planes. The intersection of a ray with the planes is then calculated, rather than the intersection of a ray with the different voxels. Determining the intersection of a ray with equally spaced, parallel planes is a simple problem. One simply needs to calculate the intersection with the first plane and the rest follow at fixed intervals because the planes are equally spaced.

To give an example of what Siddon proposes we look at the two-dimensional scenario. In two dimensions voxels become pixels and planes become lines without loss of any generality. From figure 2.7 we can see that a ray's intersections with pixels are a subset of the ray's intersection with the lines that make up the pixels. Determining that subset is straightforward.

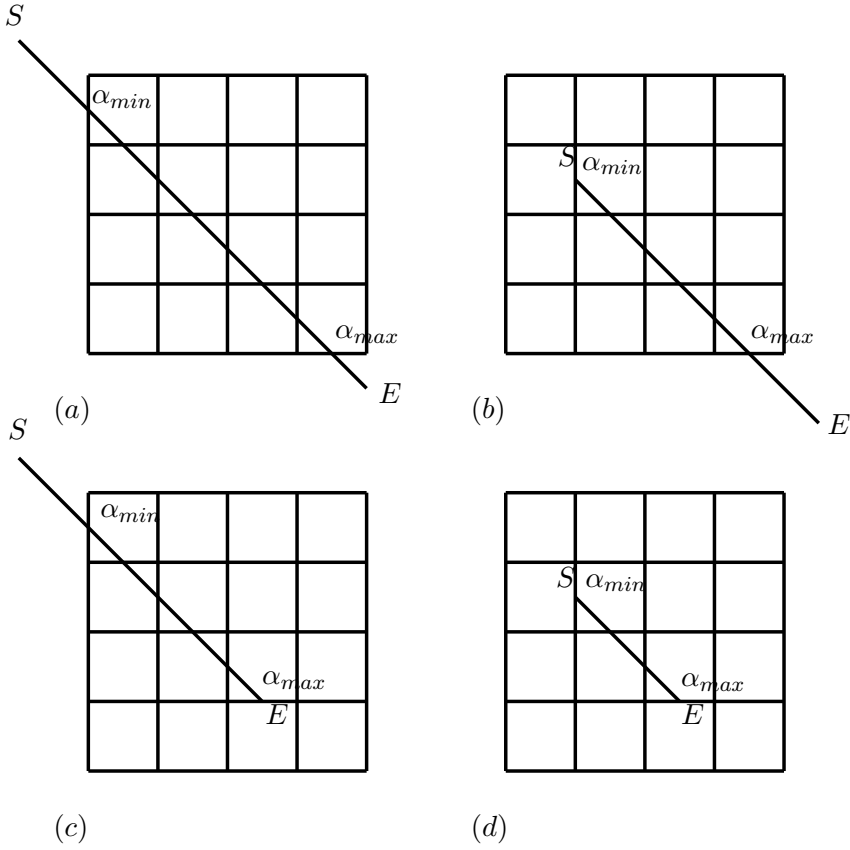


Figure 2.8: The definition of α_{min} and α_{max} . (a) Point S and E are outside the array, (b) point S is inside and point E outside the array, (c) point S is outside and point E inside the array and (d) point S and E are inside the array. Note that the points inside the array does not necessarily have to start or end on a plane.

A ray from point S to point E may be represented parametrically as

$$\begin{aligned}
 X(\alpha) &= S_x + \alpha(E_x - S_x) \\
 Y(\alpha) &= S_y + \alpha(E_y - S_y) \\
 Z(\alpha) &= S_z + \alpha(E_z - S_z) \quad .
 \end{aligned}
 \tag{2.3.1}$$

The allowed parametric range is given by α_{min} and α_{max} . All the intersections with the lines must have parametric values in the range $[\alpha_{min} \dots \alpha_{max}]$. Figure 2.8 shows how these values are obtained for different types of rays. When point S is situated inside the array as in figure 2.8(b) and figure 2.8(d), $\alpha_{min} = 0$. When point E is situated inside the array as in figure 2.8(c) and figure 2.8(d), $\alpha_{max} = 1$.

To compute the intersection of the ray with the pixels one simply determine the parametric intersection values, discard all the values that does not fall in the range $[\alpha_{min} \dots \alpha_{max}]$ and merge the three sets of parametric data into one set, creating an ordered union of the sets. The difference between two adjacent elements in this set is the length of the ray contained in a pixel. For each intersection length the corresponding pixel indices can be computed. The radiological path is then calculated as the sum of the products between the intersection lengths and the pixel attenuation value.

2.3.1.1 Siddon's algorithm

Consider a CT cube with dimensions (N_x, N_y, N_z) . These are the bounds on the indices of the data. The equations for the equally spaced, parallel planes on each axis is given by

$$\begin{aligned} X_{plane}(i) &= X_{plane}(0) + i.d_x \\ Y_{plane}(j) &= Y_{plane}(0) + j.d_y \\ Z_{plane}(k) &= Z_{plane}(0) + k.d_z \end{aligned} \quad , \quad (2.3.2)$$

where $i = 0, \dots, N_x$, $j = 0, \dots, N_y$ and $k = 0, \dots, N_z$. d_x , d_y and d_z are the distances between the x , y and z planes, respectively. α_{min} and α_{max} are obtained by calculating the intersections with the sides of the CT cube. From equations (2.3.1) and (2.3.2) we get the following:

If $E_x - S_x \neq 0$,

$$\begin{aligned} \alpha_x(0) &= \frac{X_{plane}(0) - S_x}{E_x - S_x} \\ \alpha_x(N_x) &= \frac{X_{plane}(N_x) - S_x}{E_x - S_x} \end{aligned} \quad ;$$

if $E_y - S_y \neq 0$,

$$\begin{aligned} \alpha_y(0) &= \frac{Y_{plane}(0) - S_y}{E_y - S_y} \\ \alpha_y(N_y) &= \frac{Y_{plane}(N_y) - S_y}{E_y - S_y} \end{aligned} \quad ;$$

if $E_z - S_z \neq 0$,

$$\begin{aligned} \alpha_z(0) &= \frac{Z_{plane}(0) - S_z}{E_z - S_z} \\ \alpha_z(N_z) &= \frac{Z_{plane}(N_z) - S_z}{E_z - S_z} \end{aligned} \quad .$$

If $E_x - S_x = 0$, then the ray is perpendicular to the x plane. In this case the values of α_x can be ignored in the following calculation. The same applies to

α_y and α_z . The minimum and maximum parameterised coefficients on each axis are defined as

$$\begin{aligned}\alpha_{min}^x &= \min(\alpha_x(0), \alpha_x(N_x)) \\ \alpha_{max}^x &= \max(\alpha_x(0), \alpha_x(N_x)) \quad .\end{aligned}$$

The calculations for α_{min}^y , α_{max}^y , α_{min}^z and α_{max}^z are similar. The minimum and maximum parameterised coefficients are then given by

$$\begin{aligned}\alpha_{min} &= \max(0, \alpha_{min}^x, \alpha_{min}^y, \alpha_{min}^z) \\ \alpha_{max} &= \min(1, \alpha_{max}^x, \alpha_{max}^y, \alpha_{max}^z) \quad .\end{aligned}\quad (2.3.3)$$

Note that 0 and 1 have been introduced into the computation to handle cases where the rays start or end inside the volume. If $\alpha_{min} \geq \alpha_{max}$ then the ray does not intersect with the volume. The range of indices of the planes that have intersections in the range $\alpha_{min} \leq \alpha \leq \alpha_{max}$ are given as (i_{min}, i_{max}) , (j_{min}, j_{max}) and (k_{min}, k_{max}) for their respective planes.

If $E_x - S_x \geq 0$

$$\begin{aligned}i_{min} &= N_x - \left\lfloor \frac{X_{plane}(N_x) - \alpha_{min}(E_x - S_x) - S_x}{d_x} \right\rfloor \\ i_{max} &= 1 + \left\lfloor \frac{S_x + \alpha_{max}(E_x - S_x) - X_{plane}(0)}{d_x} \right\rfloor \quad ;\end{aligned}\quad (2.3.4)$$

if $E_x - S_x < 0$

$$\begin{aligned}i_{min} &= N_x - \left\lfloor \frac{X_{plane}(N_x) - \alpha_{max}(E_x - S_x) - S_x}{d_x} \right\rfloor \\ i_{max} &= 1 + \left\lfloor \frac{S_x + \alpha_{min}(E_x - S_x) - X_{plane}(0)}{d_x} \right\rfloor \quad ,\end{aligned}\quad (2.3.5)$$

with similar equations for j_{min} , j_{max} , k_{min} and k_{max} .

The parametric intersections with the planes are given by the sets $\{\alpha_x\}$, $\{\alpha_y\}$ and $\{\alpha_z\}$ for their respective planes. The sets are constructed as follows: If $E_x - S_x > 0$,

$$\{\alpha_x\} = \{\alpha_x(i_{min}), \dots, \alpha_x(i_{max})\} \quad (2.3.6)$$

else if $E_x - S_x < 0$,

$$\{\alpha_x\} = \{\alpha_x(i_{max}), \dots, \alpha_x(i_{min})\} \quad , \quad (2.3.7)$$

where

$$\begin{aligned}\alpha_x(i) &= \frac{X_{plane}(i) - S_x}{E_x - S_x} \\ &= \alpha_x(i-1) + \frac{d_x}{E_x - S_x}\end{aligned}$$

and α_y and α_z have similar expressions.

The sets $\{\alpha_x\}$, $\{\alpha_y\}$ and $\{\alpha_z\}$ are each arranged in ascending order as indicated by equations (2.3.6) and (2.3.7). The intersections of a ray with the voxels are found by merging the three sets into one set, the result being a sorted union of sets. To cater for the scenario where the start and end point of the ray may be inside the voxel space but not located on one of the planes, the values α_{min} and α_{max} are appended to the merged sets. Let the set $\{\alpha\}$ denote the terms α_{min} , α_{max} and the merged sets $\{\alpha_x\}$, $\{\alpha_y\}$ and $\{\alpha_z\}$. That is,

$$\begin{aligned} \{\alpha\} &= \{\alpha_{min}, \text{merge}[\{\alpha_x\}, \{\alpha_y\}, \{\alpha_z\}], \alpha_{max}\} \\ &= \{\alpha_0, \dots, \alpha_n\} \quad , \end{aligned}$$

where the last index n is given by

$$n = (i_{max} - i_{min} + 1) + (j_{max} - j_{min} + 1) + (k_{max} - k_{min} + 1) + 1 \quad .$$

The adjacent terms of $\{\alpha\}$ refer to the intersection of a ray with a particular voxel. The voxel intersection length $l(m)$ for intersections m and $m-1$ is given by

$$l(m) = d_{SE} (\alpha(m) - \alpha(m-1)) \quad ,$$

where $m \in [1, \dots, n]$ and d_{SE} is the distance from point S to point E , given by

$$d_{SE} = \|SE\| = \sqrt{(E_x - S_x)^2 + (E_y - S_y)^2 + (E_z - S_z)^2} \quad . \quad (2.3.8)$$

The voxel $(i(m), j(m), k(m))$ that corresponds to intersections m and $m-1$ is the one that contains the midpoint of the two intersections. From equations (2.3.1), (2.3.2) and (2.3.3) it follows that the indices are given by

$$\begin{aligned} i(m) &= \left\lfloor \frac{S_x + \alpha_{mid} \cdot (E_x - S_x) - X_{plane}(0)}{d_x} \right\rfloor \\ j(m) &= \left\lfloor \frac{S_y + \alpha_{mid} \cdot (E_y - S_y) - Y_{plane}(0)}{d_y} \right\rfloor \\ k(m) &= \left\lfloor \frac{S_z + \alpha_{mid} \cdot (E_z - S_z) - Z_{plane}(0)}{d_z} \right\rfloor \quad , \end{aligned}$$

where α_{mid} is given by

$$\alpha_{mid} = \frac{\alpha(m) + \alpha(m-1)}{2} \quad .$$

The radiological path length d can now be written as

$$\begin{aligned} d &= \sum_{m=1}^n l(m) \rho(i(m), j(m), k(m)) \\ &= d_{SE} \sum_{m=1}^n (\alpha(m) - \alpha(m-1)) \rho(i(m), j(m), k(m)) \quad . \end{aligned}$$

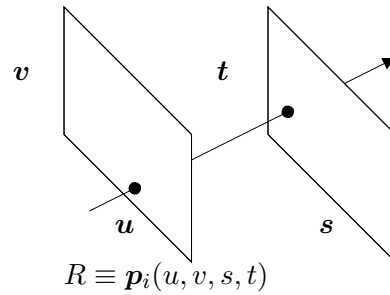


Figure 2.9: A ray in a light field coordinate system.

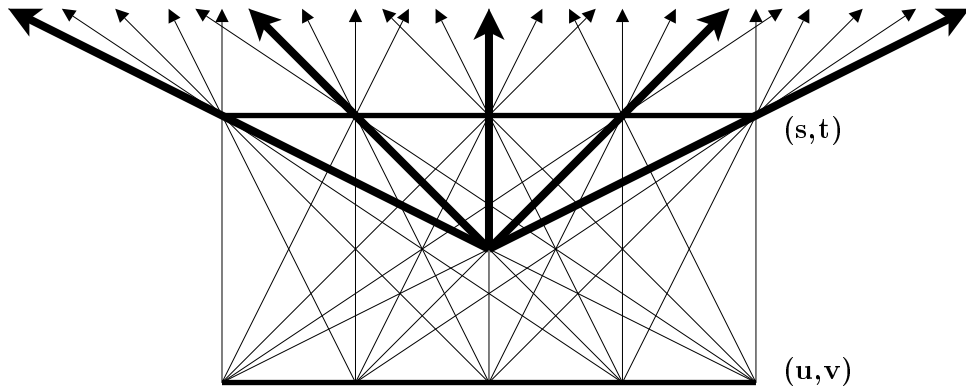


Figure 2.10: A 2D view of a light slab. A novel view (bold arrows) from within the light slab can be constructed using previously sampled values (normal arrows).

2.3.2 Light fields

Light fields is an image-order volume rendering algorithm that was originally proposed by Levoy & Hanrahan (1996). It can be described as a way of parameterising the set of all rays that emanate from a static scene. Each ray is identified by its intersection with two arbitrary planes in space. By convention the first plane is called the *focal plane* and its coordinate system is referred to as (u, v) . The second plane is called the *image plane* and its coordinates are given by (s, t) . Figure 2.9 shows a ray passing through these two planes. It follows that every ray in this space can be represented as a point or pixel value $\mathbf{p}_i = (u_i, v_i, s_i, t_i)$ in 4-dimensional space.

In practice u, v, s and t are bounded by the interval $[0 \dots 1]$. A *light slab* is the shape that is created when the focal plane and the image plane are connected. This represents all the light that enters the restricted focal plane and exits the restricted image plane.

If one can generate infinitely many rays inside a light slab, one can recreate

almost any image with a focal point or viewpoint inside the light slab. This is done by finding the associated rays and their corresponding pixel values. Figure 2.10 shows a light slab with rays passing through five sampling points on each plane. The bold rays show how a novel view can be constructed where the focal point is located at the centre of the light slab. In practice one cannot generate infinitely many rays and are thus constrained to generate a large number and compute the missing rays using some form of interpolation.

Russakoff, Rohlfing, Rueckert, Shahidi, Kim & Maurer (2003) points out three principle drawbacks when using light fields:

1. The 4D space must be densely sampled to create good images.
2. The static scene must be free of occluding objects.
3. The illumination of the scene must be fixed.

Fortunately when dealing with DRR generation only the first drawback applies. The CT data used to generate the light slab do not contain occluding objects. Only the relevant patient anatomy is scanned and converted to Hounsfield numbers. Illumination is not applicable when DRRs are generated since the radiological path lengths are used, as opposed to light or radiance which is used in general light field rendering.

2.3.3 Shear-warp factorisation

Shear-warp factorisation is an object-order rendering algorithm that has the advantage that rows of voxels in the data volume are aligned in parallel with rows of pixels in an intermediate image. This step is called shearing. An intermediate image is created using a scan line based algorithm that traverses the data volume and intermediate image in synchrony. The intermediate image then gets warped to form the final image.

The arbitrary nature of the transformation from object space to image space complicates efficient projection in object-order volume rendering algorithms. This complication can be avoided if the data volume is transformed to an intermediate coordinate system for which there is a very simple mapping from the original (object) coordinate system and which allows efficient projection. Lacroute & Levoy (1994) called this intermediate coordinate system the *sheared object space* and defined it as follows:

- By construction all viewing rays in sheared object space are parallel to the third coordinate axis.

Figure 2.11 shows the transformation to sheared object space for a parallel projection. The image shows a cross-section of the volume data (the volume slices) being sheared parallel to the set of slices that is most perpendicular to the viewing direction. This yields viewing rays that are perpendicular to the

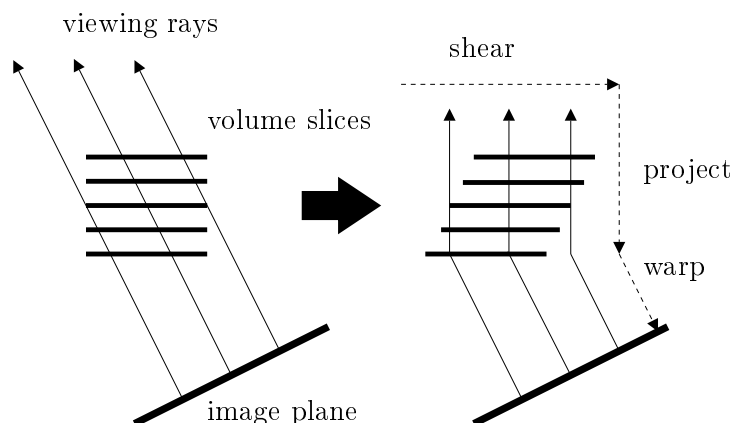


Figure 2.11: The transformation of a volume to sheared object space for a parallel projection. Note that the rays are in the viewing direction.

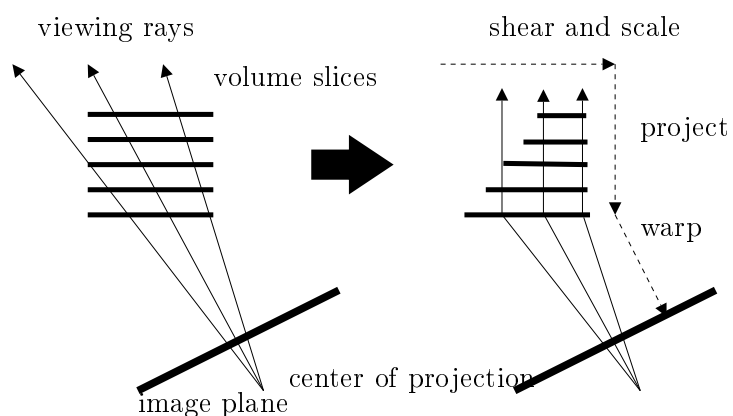


Figure 2.12: The transformation of a volume to sheared object space for a perspective projection. Each slice is translated and scaled.

volume slices. For perspective transformations the slices need to be sheared and scaled, as shown in figure 2.12.

The formal equation for transforming object coordinates into sheared coordinates is given by $M_{view} = P \cdot S \cdot M_{warp}$, where M_{view} , the viewing transformation matrix, is factorised into the following factors:

P A permutation matrix which transposes the coordinate system to make the z -axis the principle viewing axis.

S A matrix that transforms the volume into sheared object space.

M_{warp} A matrix that transforms sheared object coordinates into image coordinates.

Note that these matrices are 4×4 transformation matrices applied to homogeneous coordinates.

For general parallel projections S has the form of a shear perpendicular to the z -axis:

$$S_{par} = \begin{pmatrix} 1 & 0 & s_x & 0 \\ 0 & 1 & s_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} .$$

Here s_x and s_y are computed from the elements of M_{view} . For general perspective projections S has the form:

$$S_{persp} = \begin{pmatrix} 1 & 0 & s'_x & 0 \\ 0 & 1 & s'_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & s'_w & 1 \end{pmatrix} .$$

If one want to transform a particular slice of voxel data from object space to sheared object space for a perspective projection, then any point $\mathbf{p} = (p_x, p_y, p_z, 1)^T$ on that slice must be translated by $(p_z s'_x, p_z s'_y)$ and then scaled by $\frac{1}{1+p_z s'_w}$. It follows that

$$M_{warp} = S^{-1} \cdot P^{-1} \cdot M_{view} .$$

A simple algorithm based on the shear-warp factorisation operates as follows:

1. Transform the volume data to sheared object space. This is done by translating and resampling each slice according to S . For perspective transformations each slice is also scaled. The permutation matrix P specifies which of the three possible slicing directions to use.
2. Composite the resampled slices in front-to-back order using the "over" operator defined in Porter & Duff (1984). This generates an intermediate image in sheared object space. Note that when using this algorithm for DRR generation, the "over" operator is replaced by a function to compute the radiological path length.
3. Transform the intermediate image to image space by warping it according to M_{warp} . This produces the final image.

Doing a projection in sheared object space has the following advantages:

1. Scan lines of pixels in the intermediate image are parallel to scan lines of voxels in the data volume.
2. All voxels in a given slice are scaled by the same factor.

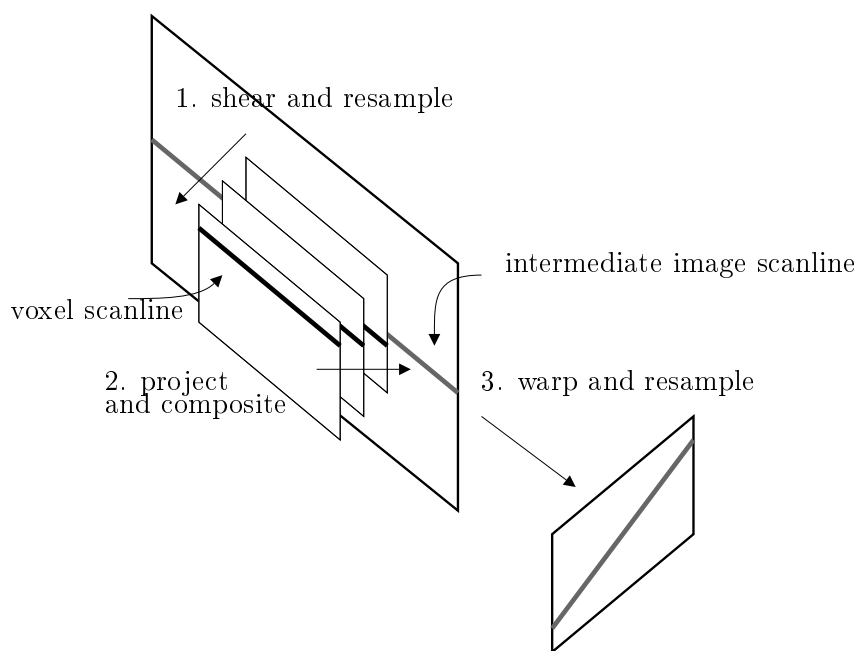


Figure 2.13: The three steps of shear-warp algorithms: shear and resample the volume slices, project resampled voxel scan lines onto intermediate image scan lines and warp the intermediate image into the final image.

3. For parallel projections all voxel slices have the same scale factor. This factor can be chosen arbitrarily. If a unity scale factor is chosen then there is a one-to-one mapping between voxels and intermediate image pixels.

2.3.3.1 Parallel projection

Shear-warp factorisation has the property that scan lines of pixels in the intermediate image are parallel to scan lines of voxels in the image data. The volume and data image data structures can thus be traversed in scan line order. Lacroute & Levoy (1994) proposes a run-length encoding of voxel scan lines which allows one to skip runs of transparent voxels.

2.3.3.2 Perspective projection

For perspective projection each slice of the volume is transformed to sheared object space by a translation and a uniform scale, whereafter the slices are resampled and composited together. No filter is necessary since the sampling rate within each slice is uniform.

This rendering method is especially suitable for 2D-3D image registration, because it can efficiently skip regions of the CT data which do not contribute to the DRR (Weese *et al.*, 1999).

2.3.4 Cylindrical harmonics

Wang, Davis & Vemuri (2002) proposes an algorithm for generating DRRs. This algorithm is based on the construction of projections of each harmonic in a cylindrical harmonic expansion of the CT data. What makes this algorithm interesting is that, after constructing a DRR from an arbitrary projection point, DRRs from other projection points can be efficiently computed. Only the weighted superpositions of the basis projects needs to be recomputed.

2.3.4.1 Definitions

Let $\rho(i, j, k)$ denote the voxel density in a 3-dimensional volume and let

$$\rho(i, j, k) = \sum_{h=1}^n a_h \phi_h(i, j, k)$$

be its expansion in cylindrical harmonics about the z axis, where $\{\phi_h(i, j, k)\}$ are the cylindrical harmonic basis functions resampled from cylindrical coordinates to the Cartesian coordinate system. The set $\{\phi_n\}$ is scaled so that $\langle \phi_m, \phi_n \rangle = \delta_{m,n}$, the Kronecker delta function. In other words, $\{\phi_h\}$ are orthonormal.

Let $\rho_\theta(i, j, k)$ denote $\rho(i, j, k)$ rotated about the z axis through an angle θ . From the properties of the expansion

$$\rho_\theta(i, j, k) = \sum_{h=1}^n e^{-in\theta} (a_h \phi_h(i, j, k)) \quad . \quad (2.3.9)$$

Thus $\rho_\theta(i, j, k)$ is the superposition of linearly phase shifted versions of the cylindrical harmonics. If θ is discretised as R uniformly distributed points in the interval $[0 \dots 2\pi)$, then equation (2.3.9) becomes

$$\rho_r(i, j, k) = \sum_{h=1}^n e^{-in\frac{2\pi r}{R}} (a_h \phi_h(i, j, k)) \quad , \quad (2.3.10)$$

where r is an index into the discrete range, so that $\theta = \frac{2\pi r}{R}$.

Let $g(u, v)$ be the projection of $\rho(i, j, k)$ for some specified parallel projection geometry and let P denote the parallel projection operator. Since P is linear it follows that

$$g(u, v) = P\rho(i, j, k) = P \sum_{h=1}^n a_h \phi_h(i, j, k) = \sum_{h=1}^n a_h \psi_h(u, v) \quad .$$

The projection of $g(u, v)$ is simply the superposition of the projections of the cylindrical harmonics of $\rho(i, j, k)$, i.e. $P\phi_h(i, j, k) = \psi(u, v)$.

The significance of this result is that the projection of $\rho_r(i, j, k)$ can be expressed as the superposition of linearly phase shifted versions of the cylindrical harmonic projections, as follows:

$$g_r(u, v) = P\rho_r(i, j, k) = \sum_{h=1}^n e^{-in\frac{2\pi r}{R}} a_h \psi_h(u, v) \quad .$$

2.3.4.2 The algorithm

The cylindrical harmonics algorithm can be broken into three distinct parts:

1. Compute the cylindrical harmonics of the given CT cube.
2. Compute projections of the cylindrical harmonics.
3. Compute projections as weighted (phase shifted) superpositions of the harmonic projections.

It is important to note that one does not need to compute all the harmonics when using the harmonic expansion. Instead, the harmonic expansion may be truncated in accordance with the desired accuracy of the DRR.

The efficiency of this algorithm is limited to rotations. Furthermore, for perspective projections, as required by this thesis, P is not linear and therefore this algorithm cannot be used.

2.3.5 Splatting or voxel projection

Splatting is an object-order algorithm. The algorithm iterates through the collection of voxels in the CT cube and projects each voxel onto an image plane, which makes up the DRR image. Splatters are also called forward projection algorithms since voxels are projected directly onto the image, in the same direction as the light rays. When doing perspective projections the footprints created on the image plane are irregular hexagons (see figure 2.14) with a maximum intensity at the centre which declines towards the boundaries. The intensities that different voxels contribute to a pixel on the image are summed. To compute a voxel footprint and the corresponding intensities of the pixels it covers for each voxel in the CT data is very expensive, since it is view dependent. The footprint must be scaled, rotated and transformed depending on the viewing transformation. In a perspective projection every voxel has its own footprint. In theory, a splatting algorithm can produce exactly the same image as a ray cast algorithm. However, it is difficult to implement an algorithm that is both efficient and that produces high quality results (Lacroute and Levoy, 1994).

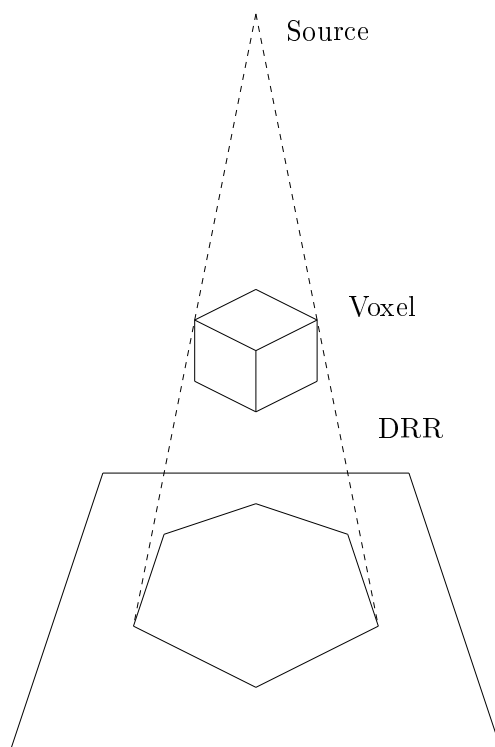


Figure 2.14: The projection of a voxel onto a plane.

2.4 Parallelisation and OpenMP

Symmetric multiprocessing, or SMP, is a multiprocessor computer architecture where two or more identical processors are connected to a single shared main memory. Most common multiprocessor systems today use an SMP architecture.

SMP systems allow any processor to work on any task no matter where the data for that task are located in memory. With proper operating system support, SMP systems can easily move tasks between processors to balance the workload efficiently.

OpenMP is described as a set of compiler directives, library routines, and environment variables that can be used to specify shared-memory parallelism in C, C++ and Fortran programs (OpenMP Architecture Review Board, 2005). It provides a model for parallel programming that is portable across shared memory architectures. OpenMP allow users to create and manage portable parallel programs. The OpenMP directives extend the C, C++ and Fortran languages with single program multiple data (SPMD) constructs, work-sharing constructs, and synchronisation constructs, and they provide support for the sharing and privatisation of data. The runtime environment can be controlled

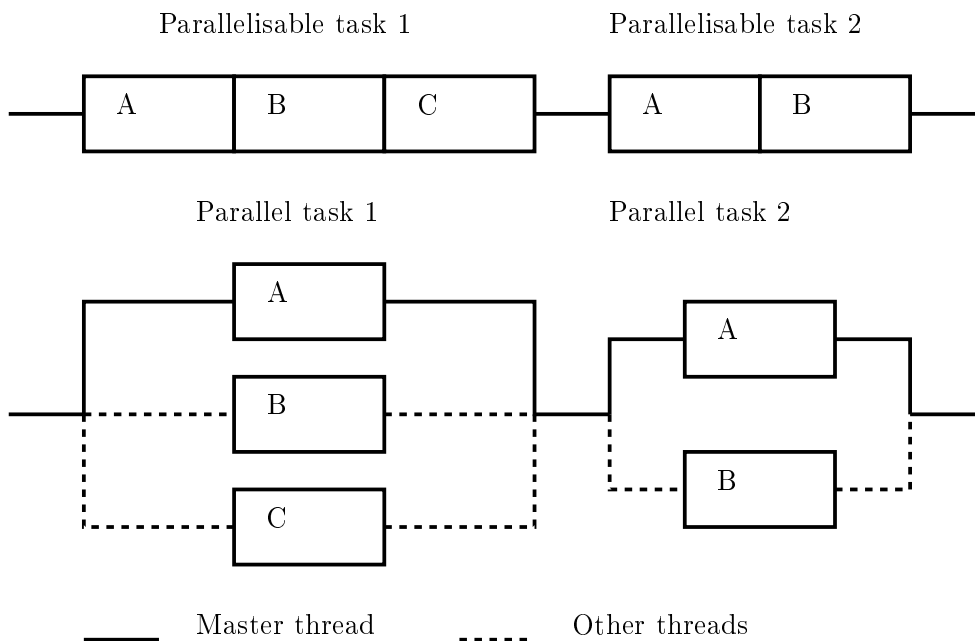


Figure 2.15: A sequential but parallelisable algorithm (top) and the equivalent algorithm using fork-join parallelism (bottom).

using the library routines and environment variables. Figure 2.15 illustrates how some algorithms can be parallelised.

An example of the trivial use of OpenMP is given by the following code snippet:

```

#ifdef _OPENMP
#pragma omp parallel for
#endif
for (int j=0; j<height; j++)
#ifdef _OPENMP
#pragma omp parallel for
#endif
  for (int i=0; i<width; i++)
    image(i,j)=raycast(focalPoint,
                       targetPoint(i,j),
                       volumeData);

```

The `omp for` directive specifies that the iterations of the associated loop will be executed in parallel. The iterations of the loop are distributed across different threads. If the `ifdef` guards are omitted and the compiler does not

support OpenMP (or it was simply not enabled), the compiler should only generate a warning message and compile the sequential code.

OpenMP has the benefit that it is extremely simple to use. The data layout and decomposition is handled automatically by the directives. No dramatic code change is needed. One needs only a single code base for both serial and parallel applications, because the compiler treats the OpenMP directives as comments if OpenMP support is not available. Furthermore, serial code statements in general need not be modified when parallelised with OpenMP. This reduces the chance of inadvertently introducing bugs. Coarse-grained and fine-grained parallelism are also possible.

On the downside OpenMP only runs efficiently on shared-memory multi-processor platforms, it requires a compiler that supports it, the scalability is limited by memory architecture, reliable error handling is missing, thread to processor mappings cannot be handled and synchronisation between a subset of threads is not allowed.

In the ideal case one expects an N times speedup when N processors are available. This is seldom the case for the following reasons:

- A large portion of the program may not be parallelised by OpenMP, which sets the theoretical upper limit of speedup according to Amdahl's law (Amdahl, 1996). For the case of parallelisation, Amdahl's law states that if F is the fraction of a program or algorithm that is sequential (i.e. cannot benefit from parallelisation), and $(1 - F)$ is the fraction that can be parallelised, then the maximum speedup that can be achieved by using N processors is $\frac{1}{F + (1-F)/N}$. As N tends to infinity the maximum speedup tends to $\frac{1}{F}$. In practice the price/performance ratio falls rapidly as N is increased once $\frac{1-F}{N}$ is small compared to F . Parallel computing is only useful when either a small number of processors is used, or when the problem has a low value of F .
- N processors may provide N times the computation power of a single processor, but the memory bandwidth usually does not scale with N . Quite often performance degradation may be observed when processors compete for shared memory bandwidth.
- Load balancing, synchronisation overhead and other common problems encountered in parallel computing also apply to OpenMP.

Carstens & Muller (2007) showed a 5-6 times performance increase can be achieved when generating ray casted DRRs using OpenMP on a computer with 8 CPUs. The same results were found in section §5.3 tables 5.3 to 5.5. Note that an improvement of approximately 8 times, which is the maximum theoretical increase, was achieved when parallelising the light field algorithm in section §5.3. The difference in the performance increase can be attributed to the fact that the ray cast algorithm has varying computation times for different

rays since it is view dependent, whereas the light field algorithm has constant computation times.

2.5 Summary

The object-order algorithms were discarded as they are not trivially parallelisable. This is true because, if multiple voxels are processed at the same time, a complex locking mechanism must be used to prevent simultaneous updates of pixel regions that overlap. This would be excessively complex and expensive. The image order algorithms, on the other hand, are trivially parallelisable and can be adapted for parallel execution with almost no overhead increase.

The ray cast algorithm, being the closest approximation to the physics involved when an X-ray is taken, was chosen to act as the gold standard of DRR generation. Also, the light field algorithm was chosen. This decision was based on the fact that the light field algorithm allows costly ray casting to be done pre-operatively when constructing the light slab and then performs quick interpolation to generate DRRs during registration. Furthermore, the problem defined in this paper requires only one light slab to be computed, as opposed to six that would have been required for arbitrary DRR generation.

Chapter 3

Improving DRR generation time

In this chapter various techniques for improving the generation time of DRRs are discussed. Section §3.1 discusses two optimisation techniques for Siddon’s algorithm and section §3.2 elaborates on how light fields can be used for DRR generation. The light field algorithm provides the benefit that costly view dependent computations are done pre-operatively. The DRR generation done intra-operatively then simply reduces to calculating interpolated values from the pre-computed samples. Section §3.3 describes how the ray cast and light field algorithms can be parallelised using OpenMP.

3.1 Ray casting: Optimised versions of Siddon’s algorithm

This section describes three types of optimisations for Siddon’s algorithm. The first optimisation uses incremental techniques to avoid costly voxel index computations and the merging of arrays. The second optimisation exploits voxel array symmetry in order to avoid recalculation of previously computed values. Finally, a method called early ray termination is presented.

3.1.1 Optimising Siddon’s algorithm through incremental techniques

An optimised version of Siddon’s algorithm was proposed by Jacobs *et al.* (1998), who claimed an improvement factor of 7.5 over Siddon’s algorithm. The proposed algorithm reduces computation by eliminating the need to explicitly compute the voxel indices i_m , j_m and k_m for every interval m . Also, it removes the need to allocate memory for the different α -arrays.

For this algorithm it is assumed that the start and end points of the ray are not located inside the CT cube. The algorithm is the same as Siddon’s algorithm until the values α_{min} and α_{max} are calculated (equations (2.3.2) to (2.3.3)). Instead of calculating the sets $\{\alpha_x\}$, $\{\alpha_y\}$ and $\{\alpha_z\}$, only the

values

$$\begin{aligned}\alpha_x &= \alpha_x(i_{min}) \\ \alpha_y &= \alpha_y(j_{min}) \\ \alpha_z &= \alpha_z(k_{min})\end{aligned}\tag{3.1.1}$$

are calculated. These are the intersections of the rays with the x -, y - and z -planes after it entered the voxel space.

To simplify some of the definitions that follow, we define the functions $\varphi_x(\alpha)$, $\varphi_y(\alpha)$ and $\varphi_z(\alpha)$, where

$$\varphi_x(\alpha) = \frac{S_x + \alpha(E_x - S_x) - b_x}{d_x} \quad ,\tag{3.1.2}$$

with $b_x = X_{plane}(0)$ from equation (2.3.2) and d_x the distance between two x -planes. The definitions of $\varphi_y(\alpha)$ and $\varphi_z(\alpha)$ are similar.

We calculate i_{min} and i_{max} using the following definitions:

$$\begin{aligned}\alpha_{min} = \alpha_{min}^x &\rightarrow i_{min} = 1 \\ \alpha_{min} \neq \alpha_{min}^x &\rightarrow i_{min} = \lceil \varphi_x(\alpha_{min}) \rceil \\ \alpha_{max} = \alpha_{max}^x &\rightarrow i_{max} = N_x \\ \alpha_{max} \neq \alpha_{max}^x &\rightarrow i_{max} = \lfloor \varphi_x(\alpha_{max}) \rfloor \\ \alpha_{min} = \alpha_{min}^x &\rightarrow i_{max} = N_x - 1 \\ \alpha_{min} \neq \alpha_{min}^x &\rightarrow i_{max} = \lfloor \varphi_x(\alpha_{min}) \rfloor \\ \alpha_{max} = \alpha_{max}^x &\rightarrow i_{min} = 0 \\ \alpha_{max} \neq \alpha_{max}^x &\rightarrow i_{min} = \lceil \varphi_x(\alpha_{max}) \rceil \quad .\end{aligned}$$

The definitions for the j and k indices are similar. These values are used to compute the number of plane crossings N_p as the ray passes through voxel space. The number of plane crossings is given by

$$N_p = (i_{max} - i_{min} + 1) + (j_{max} - j_{min} + 1) + (k_{max} - k_{min} + 1) \quad .$$

Note that N_p is actually an upper bound for the number of plane crossings, since simultaneous crossings of two or three planes are counted separately. The index (i, j, k) of the first voxel is calculated by

$$\begin{aligned}i &= \left\lceil \varphi_x \left(\frac{\min(\alpha_x, \alpha_y, \alpha_z) + \alpha_{min}}{2} \right) \right\rceil \\ j &= \left\lceil \varphi_y \left(\frac{\min(\alpha_x, \alpha_y, \alpha_z) + \alpha_{min}}{2} \right) \right\rceil \\ k &= \left\lceil \varphi_z \left(\frac{\min(\alpha_x, \alpha_y, \alpha_z) + \alpha_{min}}{2} \right) \right\rceil \quad .\end{aligned}\tag{3.1.3}$$

As we follow the ray through voxel space we have to update the values α_x , α_y , α_z , i , j and k depending on which plane is crossed. To facilitate this we compute parameterised increments between the planes on each axis. For the x -axis we would compute

$$\alpha_{xu} = \frac{d_x}{E_x - S_x} \quad .$$

The computation of α_{yu} and α_{zu} are similar. Furthermore, we have to compute the voxel index increments i_u , j_u and k_u .

$$i_u = \begin{cases} 1 & \text{if } S_x < E_x \\ -1 & \text{otherwise.} \end{cases}$$

The definitions of j_u and k_u are similar.

If $\alpha_x < \alpha_y \leq \alpha_z$ then the next intersected plane is an x -plane. The algorithm repeatedly finds the next intersected plane and updates the necessary variables to continue the next iteration. To begin the iteration we initialise a parametric position variable $\alpha_c = \alpha_{min}$ and a radiological path integral counter $d = 0$ and the α counters as shown in equation (3.1.1). While $\alpha_c < \alpha_{max}$ we repeatedly run through the following algorithm:

- If $\alpha_x \leq \alpha_y \leq \alpha_z$, then

$$l(i, j, k) \leftarrow (\alpha_x - \alpha_c)d_{SE} \quad (3.1.4)$$

$$d \leftarrow d + l(i, j, k)\rho(i, j, k) \quad (3.1.5)$$

$$i \leftarrow i + i_u$$

$$\alpha_c \leftarrow \alpha_x$$

$$\alpha_x \leftarrow \alpha_x + \alpha_{xu} \quad .$$

- If $\alpha_y \leq \alpha_x \leq \alpha_z$, then

$$l(i, j, k) \leftarrow (\alpha_y - \alpha_c)d_{SE} \quad (3.1.6)$$

$$d \leftarrow d + l(i, j, k)\rho(i, j, k) \quad (3.1.7)$$

$$j \leftarrow j + j_u$$

$$\alpha_c \leftarrow \alpha_y$$

$$\alpha_y \leftarrow \alpha_y + \alpha_{yu} \quad .$$

- If $\alpha_z \leq \alpha_x \leq \alpha_y$, then

$$l(i, j, k) \leftarrow (\alpha_z - \alpha_c)d_{SE} \quad (3.1.8)$$

$$d \leftarrow d + l(i, j, k)\rho(i, j, k) \quad (3.1.9)$$

$$k \leftarrow k + k_u$$

$$\alpha_c \leftarrow \alpha_z$$

$$\alpha_z \leftarrow \alpha_z + \alpha_{zu} \quad .$$

When the intersection lengths $l(i, j, k)$ are not explicitly needed by an application the calculations can be simplified by incorporating equation (3.1.4) into equation (3.1.5), equation (3.1.6) into equation (3.1.7) and equation (3.1.8) into equation (3.1.9), respectively, without the multiplication of d by d_{SE} (equation (2.3.8)), which can be done afterwards.

Note that this algorithm works correctly when α_x , α_y and α_z are equal to each other. Consider the case where a ray enters a CT cube on a voxel corner and exits on the opposite corner. It is important to note that the first voxel has the index $(0, 0, 0)$. Here we will have a situation where

$$\alpha_x = \alpha_y = \alpha_z > \alpha_c = \alpha_{min} \quad .$$

The initial voxel indices computed using equation (3.1.3) are $i = j = k = 0$. In the first iteration of the algorithm, any one of the α values can be used. If α_x is used, then we have

$$\begin{aligned} l(i, j, k) &\leftarrow (\alpha_x - \alpha_c)d_{SE} \neq 0 \\ i &\leftarrow i + i_u = 1 \\ \alpha_c &\leftarrow \alpha_x \\ \alpha_x &\leftarrow \alpha_x + \alpha_{xu} > \alpha_y = \alpha_z \quad . \end{aligned}$$

The contribution of voxel $(0, 0, 0)$ is not zero, since $(\alpha_x - \alpha_c) \neq 0$. The new voxel index is $(1, 0, 0)$. In the next iteration either α_y or α_z can be used. If α_y is used we have

$$\begin{aligned} l(i, j, k) &\leftarrow (\alpha_y - \alpha_c)d_{SE} = 0 \\ j &\leftarrow j + j_u = 1 \\ \alpha_c &\leftarrow \alpha_y \\ \alpha_y &\leftarrow \alpha_y + \alpha_{yu} = \alpha_x > \alpha_z \quad . \end{aligned}$$

The contribution of voxel $(1, 0, 0)$ to the radiological path is zero, since $(\alpha_y - \alpha_c) = 0$. The current voxel index is $(1, 1, 0)$. In the next iteration α_z will be used.

$$\begin{aligned} l(i, j, k) &\leftarrow (\alpha_z - \alpha_c)d_{SE} = 0 \\ k &\leftarrow k + k_u = 1 \\ \alpha_c &\leftarrow \alpha_z \\ \alpha_z &\leftarrow \alpha_z + \alpha_{zu} = \alpha_x = \alpha_y \quad . \end{aligned}$$

The contribution of voxel $(1, 1, 0)$ to the radiological path is zero, since $(\alpha_z - \alpha_c) = 0$. The current voxel index is $(1, 1, 1)$. In the next iteration any one of the α values can be used. If α_x is used, then we have

$$\begin{aligned} l(i, j, k) &\leftarrow (\alpha_x - \alpha_c)d_{SE} \neq 0 \\ i &\leftarrow i + i_u = 1 \\ \alpha_c &\leftarrow \alpha_x \\ \alpha_x &\leftarrow \alpha_x + \alpha_{xu} > \alpha_y = \alpha_z \quad . \end{aligned}$$

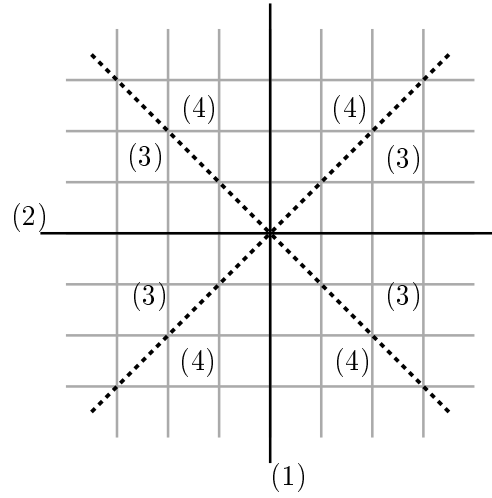


Figure 3.1: The four types of lines found when observing the geometric symmetry of a ray crossing through pixel space. Types (1) and (2) run parallel to the y and x axes, respectively. Type (3) has a slope with a magnitude less than $|\frac{d_y}{d_x}|$ and type (4) has a slope with a magnitude larger than $|\frac{d_y}{d_x}|$.

The contribution of voxel $(1, 1, 1)$ is not zero, since $(\alpha_x - \alpha_c) \neq 0$. The algorithm continues in a similar way as described above, with voxels $(2, 1, 1)$ and $(2, 2, 1)$ not contributing to the radiological path length. Voxel $(2, 2, 2)$ does contribute to radiological path length and the algorithm repeats.

3.1.2 Utilising voxel array symmetry properties

Zhao & Reader (2003) proposes a novel algorithm utilising the voxel array symmetry properties when a ray passes through a voxel space. The two-dimensional case is described first and then generalised to three dimensions.

3.1.2.1 The 2D projection algorithm

As figure 3.1 indicates, a ray passing through a pixel array can always be classified geometrically into one of four types, depending on the slope. The first two types run parallel to the x and y axes, respectively. The third type has a slope $m = \frac{E_y - S_y}{E_x - S_x}$ where $|m| \leq |\frac{d_y}{d_x}|$ and the fourth type has $|\frac{d_y}{d_x}| \leq |m|$, where d_x and d_y refer the pixel width and height. Note that $|m| = |\frac{d_y}{d_x}|$ can be handled by both the third and fourth types. The first two types are trivial cases. The third and fourth types are symmetrical and therefore only one of these types will be discussed.

Figure 3.2 is an example of a ray that passes through pixel space. It is clear that the intersection length between any two neighbouring y -planes (TL

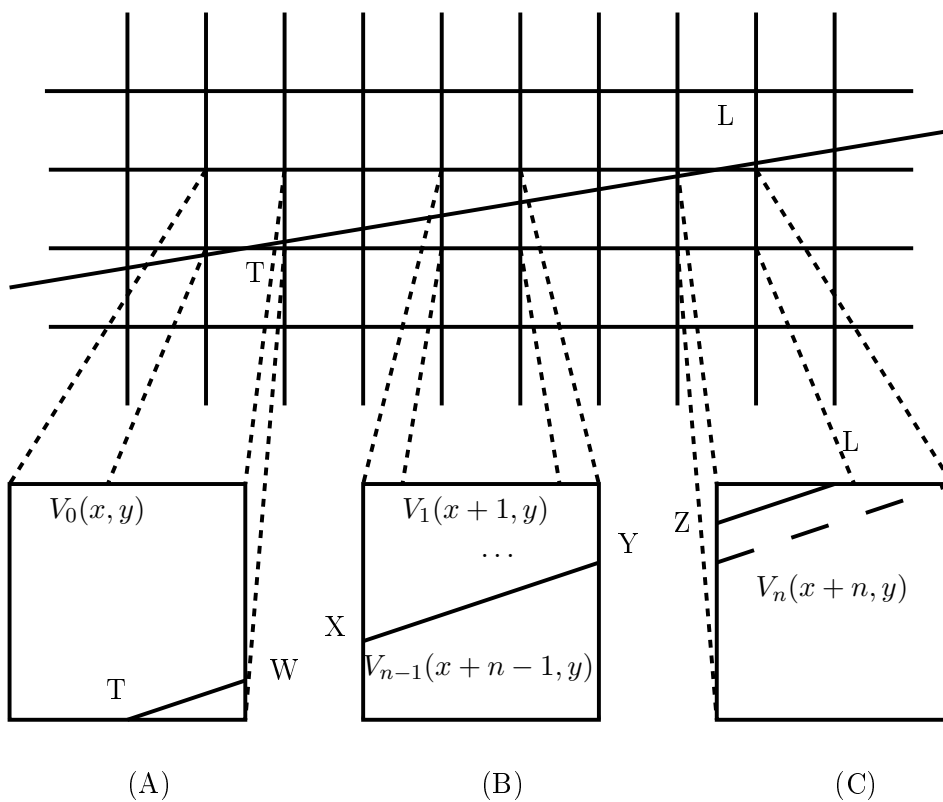


Figure 3.2: The different patterns emerging when a ray crosses through pixel space.

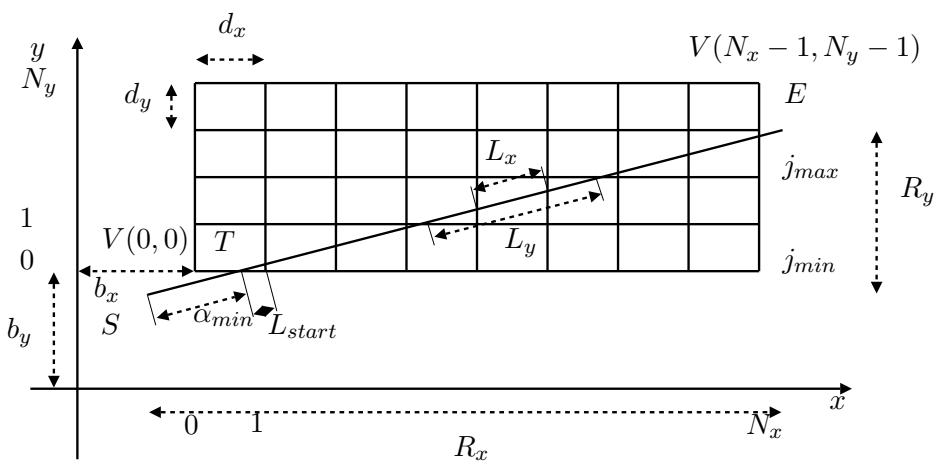


Figure 3.3: Notations used in the 2D description.

in this example) is constant. Also, the ray path always runs through a similar pattern, which is broken into sub-patterns A , B and C in figure 3.2. The patterns are described as follows:

Sub-pattern A This sub-pattern starts from a position T on the lower y -plane. The length of TW can be obtained from either the position of T and the angle of the ray, or from the last calculation in sub-pattern C (see below). Thereafter, due to the angle limitation, the ray will generally pass through the right x -plane from the current pixel $v_0(x, y)$ to the next pixel $v_1(x+1, y)$. The remaining uncalculated length becomes $r = \|TL\| - \|TW\|$.

Sub-pattern B If, for pixels $v_1(x+1, y)$ to $v_{n-1}(x+n-1, y)$, the remaining length r is longer than the intersection length between neighbouring x -planes (labelled XY), this ray will cross the pixel from the left x -plane to the right x -plane of the pixel v_1 . This case may repeat several times. After calculation of sub-pattern B, the remaining length is $r = \|TL\| - \|TW\| - (n-1)\|XY\|$, where $n-1$ is the total number of pixels meeting the requirements of sub-pattern B in this iteration. The pixel indices always increase along the x direction, e.g. $v_2(x+2, y), v_3(x+3, y), \dots, v_{n-1}(x+n-1, y)$.

Sub-pattern C When the remaining length r is shorter than $\|XY\|$, it implies that the ray will not reach the next x -plane. Rather, it will intersect with the next y -plane and the intersection length of this last pixel $v_n(x+n, y)$ is r (line segment ZL in figure 3.2). The pixel indices will be updated to $(x+n, y+1)$. In the event that r is equal to $\|XY\|$ the pixel indices will become $(x+n+1, y+1)$ (see the dashed line in figure 3.2). Furthermore, the intersection length of the first pixel in the next iteration is given by the difference between $\|XY\|$ and r , with the exception that the intersection length is $\|XY\|$ if $r = \|XY\|$. From here the pattern returns to sub-pattern A.

Let us assume we are tracing a ray SE through a pixel array of (N_x, N_y) pixels. The distances between the start and end points (refer to figure 3.3) are given by

$$\begin{aligned} R_x &= E_x - S_x \\ R_y &= E_y - S_y \quad . \end{aligned}$$

We assume that both S and E lie outside the pixel array and that $|R_x| > d_x$ and $|R_y| > d_y$. The description will be limited to the case where $|\frac{R_y}{R_x}| < |\frac{d_y}{d_x}|$, $R_x > 0$ and $R_y > 0$. The other cases are similar and follow logically using projection symmetry properties.

To compute the first intersection point $T(T_x, T_y)$ of the ray with the first y -plane of the array, the algorithm starts similar to Siddon's algorithm¹:

$$\begin{aligned}\alpha_x(0) &= \frac{b_x - S_x}{R_x} \\ \alpha_y(0) &= \frac{b_y - S_y}{R_y} \\ \alpha_x(N_x) &= \frac{b_x + N_x d_x - S_x}{R_x} \\ \alpha_y(N_y) &= \frac{b_y + N_y d_y - S_y}{R_y} .\end{aligned}$$

Here b_x and b_y are the offsets of the CT cube data in the CT coordinate system (refer to equation (3.1.2)).

$$\begin{aligned}\alpha_{min} &= \max(\alpha_x(0), \alpha_y(0)) \\ \alpha_{max} &= \min(\alpha_x(N_x), \alpha_y(N_y)) \\ j_{min} &= N_y - \left\lfloor \frac{b_y + N_y d_y - (S_y + \alpha_{min} R_y)}{d_y} \right\rfloor \\ j_{max} &= \left\lfloor \frac{S_y + \alpha_{max} R_y - b_y}{d_y} \right\rfloor \\ T_x &= S_x + \left(\frac{b_y + j_{min} d_y - S_y}{R_y} \right) R_x\end{aligned}$$

The intersection lengths between neighbouring x -planes and y -planes are then computed:

$$\begin{aligned}L_{SE} &= \|SE\| = \sqrt{R_y^2 + R_x^2} \\ L_x &= \frac{L_{SE} d_x}{R_x} \\ L_y &= \frac{L_{SE} d_y}{R_y} .\end{aligned}$$

If:

- $T_x - b_x > \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor d_x$, then

$$\|TW\| = \frac{\left(d_x - \left(T_x - b_x - \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor d_x \right) \right) L_{SE}}{R_x} .$$

- $T_x - b_x = \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor d_x$, then

$$\|TW\| = L_x .$$

¹The equations differ slightly from the ones specified in section §2.3.1.1 because of the assumptions made for the description of the algorithm.

After this initial step, the algorithm starts tracing the ray between j_{min} and j_{max} . The section between any two neighbouring y -planes is the pattern described in figure 3.2. The indices of the start voxel $V_0(v_x, v_y)$ are given by

$$\begin{aligned} v_x &= \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor \\ v_y &= j_{min} \quad . \end{aligned}$$

The first voxel length will be that of sub-pattern A:

$$\begin{aligned} l_0(v_x, v_y) &= \|TW\| \\ r_0 &= L_y - \|TW\| \quad . \end{aligned}$$

The following $n-1$ voxels which meet the condition of sub-pattern B is defined recursively as:

$$\begin{aligned} l_m(v_x + m, v_y) &= L_x \\ r_m &= r_{m-1} - L_x \quad . \end{aligned}$$

This definition is repeated until $r_n < L_x$. The last voxel is as described in sub-pattern C:

$$l_n(v_x + n, v_y) = r_n \quad .$$

The start value for the next y -level is given by:

$$\|TW\| = \begin{cases} L_x - r_n & \text{if } L_x > r_n \\ L_x & \text{if } L_x = r_n \end{cases} \quad .$$

Also, the index in the y -direction is increased to continue with the next loop.

When $i_{min} > 0$ or $i_{max} < N_x$, the algorithm has to do extra loops:

- If $i_{min} > 0$, the first voxel indices and r is given by

$$\begin{aligned} v_x &= \begin{cases} \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor & \text{if } T_x - b_x > \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor d_x \\ \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor - 1 & \text{if } T_x - b_x = \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor d_x \end{cases} \\ v_y &= j_{min} - 1 \\ r &= L_{SE} \left(\frac{b_y + j_{min}d_y - S_y}{R_y} - \alpha_{min} \right) \quad . \end{aligned}$$

- If $i_{max} < N_x$, the last voxel indices and r is given by

$$\begin{aligned} v_x &= \left\lfloor \frac{L_x - b_x}{d_x} \right\rfloor \\ v_y &= j_{max} \\ r &= L_{SE} \left(\frac{b_y + j_{min}d_y - S_y}{R_y} - \alpha_{min} \right) \quad . \end{aligned}$$

This completes the ray-tracing for the 2D case.

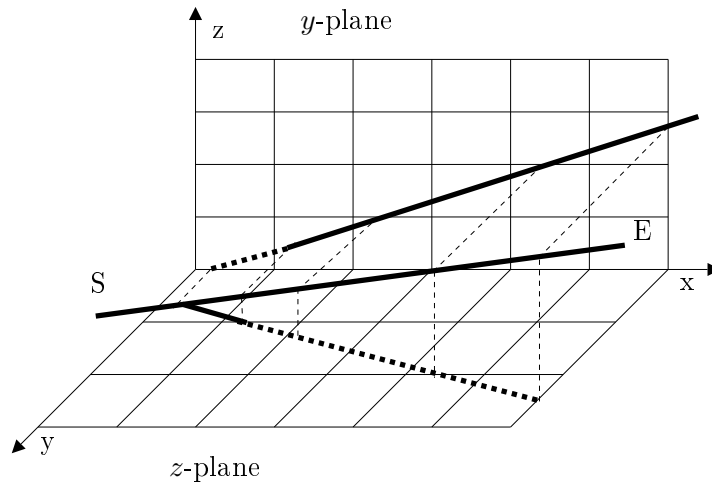


Figure 3.4: Interleaving.

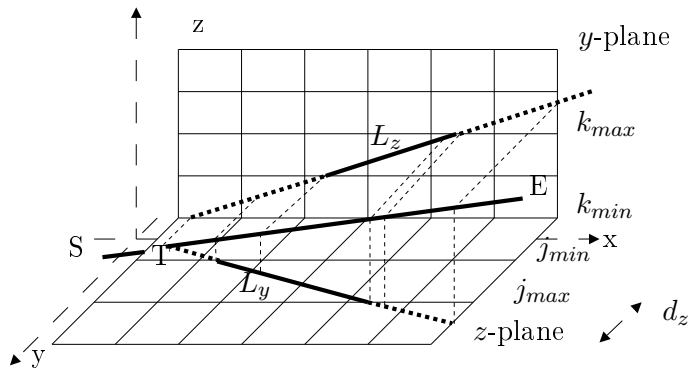


Figure 3.5: The notation for the 3D ray-tracing algorithm.

3.1.2.2 The 3D projection algorithm

The three-dimensional algorithm is similar to the two-dimensional one, but one must take into account that the ray is extending into the z -direction as well. A simple method to incorporate the extra dimension while retaining most of the two-dimensional algorithm is to convert the three-dimensional ray into two two-dimensional interleaving rays — the one being a projection on the y -parametric plane and the other on the z -parametric plane. A projection plane is simply a 2D case where, for instance, the z parametric plane is traversed while only considering movements in the x and y directions. Figure 3.4 shows a ray broken into two parts. The upper diagram is the projection onto the y -plane and the lower diagram is the projection onto the z -plane. The fine

dashed line shows when the ray crosses through any y - or z -plane. This ray is decomposed into two two-dimensional interleaving projection operations. If the y -plane is reached before the z -plane, then the algorithm will perform the tracing just between the x - and y -directions. This is simply a two-dimensional trace. Similarly, if the ray reaches a z -plane first, a two-dimensional trace is done in the x - and z -directions.

Similar to the two-dimensional case, let us assume we are tracing a ray SE through a voxel array of (N_x, N_y, N_z) voxels. The distances between the start- and end points are given by

$$\begin{aligned} R_x &= E_x - S_x \\ R_y &= E_y - S_y \\ R_z &= E_z - S_z \quad . \end{aligned}$$

We assume that both S and E lie outside the voxel array and that $|R_x| > d_x$, $|R_y| > d_y$ and $|R_z| > d_z$. The description will be limited to the case where the following conditions hold:

$$\begin{aligned} \left| \frac{R_y}{R_x} \right| &\leq \left| \frac{d_y}{d_x} \right| \\ \left| \frac{R_z}{R_x} \right| &\leq \left| \frac{d_z}{d_x} \right| \\ R_y R_x &> 0 \\ R_z R_x &> 0 \quad . \end{aligned}$$

The other cases are similar and follows logically using array symmetry properties. The starting point in the three-dimensional algorithm is the first intersection point closest to S between the ray and the y - or z -plane. The computation of $\alpha_x(0)$, $\alpha_y(0)$, $\alpha_x(N_x)$ and $\alpha_y(N_y)$ are the same as for the two-dimensional algorithm. It is extended with definitions for $\alpha_z(0)$ and $\alpha_z(N_z)$:

$$\begin{aligned} \alpha_z(0) &= \frac{b_z - S_z}{R_z} \\ \alpha_z(N_z) &= \frac{b_z + N_z d_z - S_z}{R_z} \quad , \end{aligned}$$

where $R_z = E_z - S_z$. The definitions of α_{min} and α_{max} are extended to make provision for the new dimension:

$$\begin{aligned} \alpha_{min} &= \max(\alpha_x(0), \alpha_y(0), \alpha_z(0)) \\ \alpha_{max} &= \min(\alpha_x(N_x), \alpha_y(N_y), \alpha_z(N_z)) \quad . \end{aligned}$$

The algorithm then computes the parametric planes index range j_{min} and j_{max} as described in the two-dimensional case. It also computes the index range for

the z -planes:

$$\begin{aligned} k_{min} &= N_z - \left\lfloor \frac{b_z + N_z d_z - (S_z + \alpha_{min} R_z)}{d_z} \right\rfloor \\ k_{max} &= \left\lfloor \frac{S_z + \alpha_{max} R_z - b_z}{d_z} \right\rfloor . \end{aligned}$$

Next, the intersection point T between the ray and the first y - or z -plane is calculated. If the first intersection is with a y -plane, then the location is given by

$$T_x = S_x + \left(\frac{b_y + j_{min} d_y - S_y}{d_y} \right) d_x ,$$

otherwise, the location is given by

$$T_x = S_x + \left(\frac{b_z + k_{min} d_z - S_z}{d_z} \right) d_x .$$

The parametric intersection lengths between neighbouring planes on the same axis are also calculated as in the two-dimensional case with the intersection length of the z -plane given by

$$L_z = L_{SE} \frac{d_z}{R_z} .$$

The total length SE for the three-dimensional case is calculated as

$$L_{SE} = \|SE\| = \sqrt{R_x^2 + R_y^2 + R_z^2} .$$

When T is located inside a parametric plane or voxel, the intersection length between the ray and this voxel is given by

$$\|TW\| = \frac{d_x - \left(T_x - b_x - \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor d_x \right) L_{SE}}{R_x} . \quad (3.1.10)$$

If T lies in the y - and x -plane, or the z - and x -plane, the intersection length is

$$\|TW\| = L_x .$$

The indices of the starting voxel $V_0(v_x, v_y, v_z)$ are given by the following equations:

- If T is on the y -plane

$$\begin{aligned} v_x &= \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor \\ v_y &= j_{min} \\ v_z &= k_{min} - 1 . \end{aligned}$$

- If T is on the z -plane

$$\begin{aligned} v_x &= \left\lfloor \frac{T_x - b_x}{d_x} \right\rfloor \\ v_y &= j_{min} - 1 \\ v_z &= k_{min} \quad . \end{aligned}$$

The final step of preparatory work is to calculate the distances from the start of a whole voxel boundary to the y - and z -parametric planes respectively:

$$\begin{aligned} r_y &= \left(\frac{b_y + j_{min}d_y - S_y}{R_y} - \alpha_{min} \right) L_{SE} \\ r_z &= \left(\frac{b_z + j_{min}d_z - S_z}{R_z} - \alpha_{min} \right) L_{SE} \quad . \end{aligned}$$

The main tracing loop of the algorithm starts now, ranging between the first and last indices of the y - and z -parametric planes. The values r_y and r_z are repeatedly compared. The shortest distance indicates in which two-dimensional case to trace, since there would be no interference from the other parametric plane. For example, in the case of $r_y < r_z$, the algorithm looks as follows:

1. $r_z \leftarrow r_z - r_y$
2. If $L_y < \|TW\|$, with $\|TW\|$ referring to the remaining distance to the next x -plane, then
 - a) Calculate the current length value with r_y .
 - b) $v_y \leftarrow v_y + 1$
 - c) $\|TW\| \leftarrow \|TW\| - r_y$
 - d) $r_y \leftarrow L_y$
- else
 - a) Calculate the current voxel.
 - b) $r_y \leftarrow r_y - \|TW\|$
 - c) Perform steps similar to the the 2D case.
 - d) Update the parametric index.
 - e) $r_y \leftarrow L_y$

The calculations of the initial and end phase are the same as for the two-dimensional case because there is no interference from the other parametric plane.

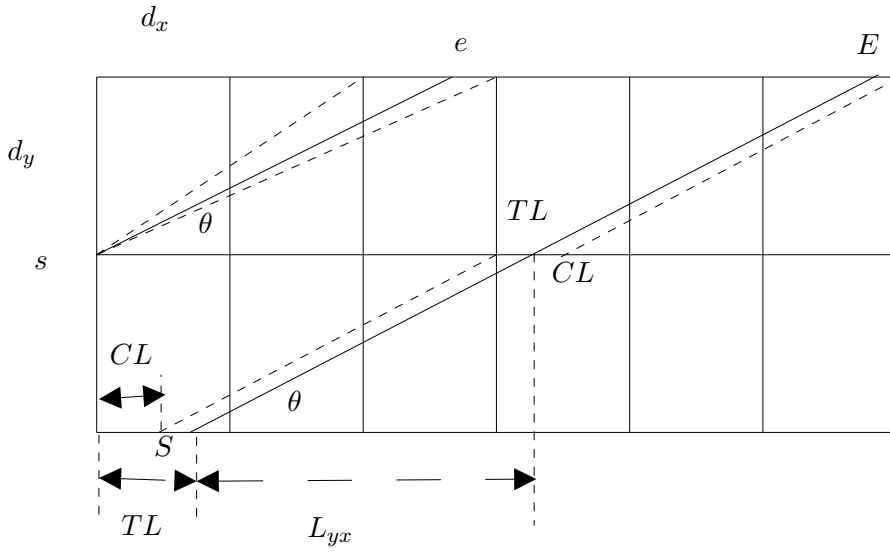


Figure 3.6: When $\|TL\| < \|CL\|$, the fragment of the ray SE will pass just one B sub-pattern. When $\|TL\| > \|CL\|$, the fragment will pass two B sub-patterns. The upper-left diagram shows the angle range of such a type of line. Note that se is parallel to SE .

Optimisations When the angle $\theta = \arctan\left(\frac{R_y}{R_x}\right)$ is close to the angle of $\arctan\left(\frac{d_y}{d_x}\right)$, the tracing slows down because the number of occurrences of sub-pattern B is very limited. A possible solution is that, every time the ray crosses a y -plane, the algorithm compares the lengths of CL and TL (refer to figure 3.6), where $\|TL\|$ is the distance from the y -plane intersect of a ray with a voxel to the first x -plane of the voxel. $\|CL\|$ is the x -offset in the voxel that would cause the ray to cross the next y -plane and an x -plane simultaneously. L_{yx} is the x -distance between y -plane intersections. If $\|CL\| < \|TL\|$ the next tracing sub-pattern sequence will be A, B, B, C . Otherwise, the sequence will be A, B, C . The updated value of $\|TL\|$ is carried over at the end of each pattern procedure.

3.1.3 Early ray termination

Early ray termination is a technique used with ray casting where a ray can be terminated when the accumulated opacity is close to unity (Meißner *et al.*, 2000). When this condition occurs further voxel examination can be skipped. Efficient implementation of early ray termination is trivial for the ray casting algorithm, but difficult for an object-order algorithm (Lacroute, 1995).

3.2 Light fields

Light fields is a simple method to construct novel views from arbitrary camera positions. This is achieved by resampling a set of existing images and is therefore called *image-based rendering* (Levoy and Hanrahan, 1996). Image-based rendering algorithms have the advantages that

- they are suitable for real-time implementations since they are not computationally expensive,
- the cost of generating a scene is not dependent on the complexity of the scene, and
- the set of base images can be real images, artificially created ones or both.

The amount of light travelling along any arbitrary ray in space is called its *radiance*². For any arbitrary scene with static illumination the radiance of all rays is called the *plenoptic function*. In the plenoptic function rays are represented by the coordinates x, y, z and the angles θ and ϕ , as seen in figure 3.7. Each ray has an associated radiance value. For objects that are concave, some rays can travel from the object and be blocked again by the object. There is no device that can measure this radiance (Levoy, 2006). If we focus only on the convex hull of the object, then we can measure the radiance using a digital camera. Also, the radiance along a ray does not change, so the 5D plenoptic function can be reduced to a 4D function. This 4D function is the formal definition of a light field.

As mentioned in section §2.3.2, a light field can be parameterised by two restricted planes, called a *light slab*. A light slab represents all the light that enters the restricted focal plane and exits the restricted image plane and every ray in this space can be represented as a point or pixel value $\mathbf{p}_i = (u_i, v_i, s_i, t_i)$ in 4D space. Let (u, v) denote a point on the focal plane and (s, t) a point on the image plane. Furthermore, let the planes be perpendicular to the x axis with the origin located in the focal plane and the image plane located at a distance f away from the origin. The 4D representation then relates to the 5D representation in the following manner:

$$\begin{aligned} (x, y, z) &= (0, u, v) \\ \tan \phi &= \frac{t - v}{s - u} \\ \tan \theta &= \frac{\sqrt{(s - u)^2 + (t - v)^2}}{f} . \end{aligned}$$

²Radiance is measured in Watts (W) per steradian (sr) per meter squared (m^2). Refer to section §B.5 for a definition of a steradian.

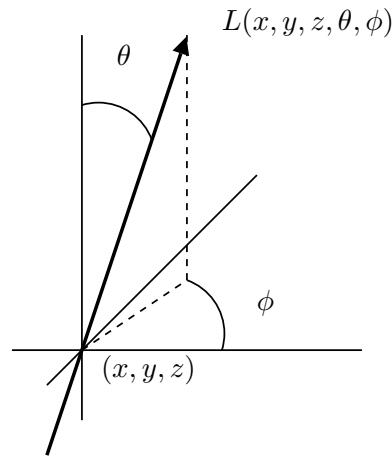


Figure 3.7: The 5D representation of a ray

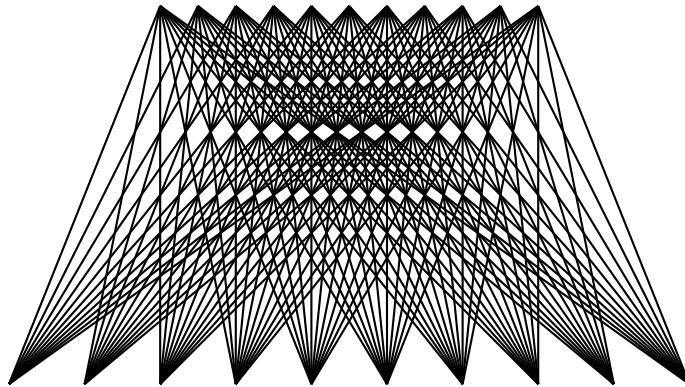


Figure 3.8: A 2D view of the rays contained in a single light slab.

This relationship is defined using the (u, v) coordinates, which are located on the focal plane where $x = 0$ and the (s, t) coordinates, which are located on the image plane where $x = f$. Note that the choice of having the focal plane at $x = 0$ is arbitrary. The planes can be located anywhere on the x -axis.

The light slab is also called a *plane-plane* representation. This type of parameterisation does not include, for instance, rays that are parallel to the two planes as can be seen in figure 3.8. However, multiple light slabs can be used to represent these rays as illustrated in figure 3.9.

It is evident from figure 3.9 that the corner points are sampled more densely. In an attempt to achieve uniform sampling, other types of parameterisations have been investigated. These parameterisations include, but are not limited to *point and direction* (Ihm *et al.*, 1997), *two sphere* and *direction and point* (Camahort *et al.*, 1998). Ihm, Park & Lee (1997) notes that in gen-

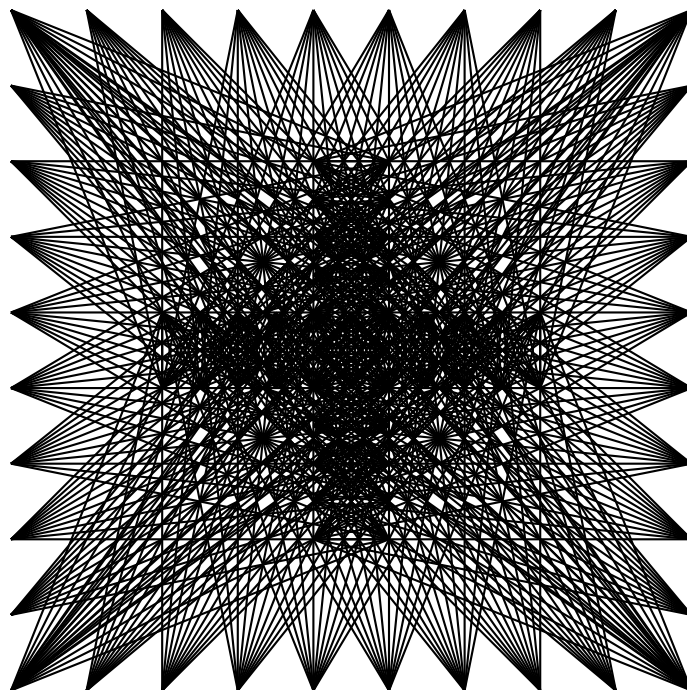


Figure 3.9: A 2D view of a combination of four light slabs used to produce a 360° view of an object located in the centre. In 3D six light slabs would be required in order to recreate any arbitrary view of an object.

eral spheres are less efficient than planes from a computational point of view. Baumann (2004) also concludes that the light slab or plane-plane parameterisation is the best for DRR generation because it provides better rendering performance than the other parameterisations. Furthermore, it is important to note that since the projection space for our DRRs is constrained, only a single light slab is necessary to represent the sampling space. Figure 3.10 shows how a light slab can be viewed as a 2D array of 2D images where the (u, v) coordinates identify a sub-image in the light slab and the (s, t) coordinates identify a pixel in the sub-image.

What makes light fields attractive for the DRR generation problem is the fact that most computation can be done pre-operatively. During patient treatment, when computation time must be minimised, it is then possible to quickly generate images from the pre-computed data. The generation of images is achieved by interpolation of the pre-computed data. This can be done in constant time, since the computation time is not dependent on the complexity of the image.

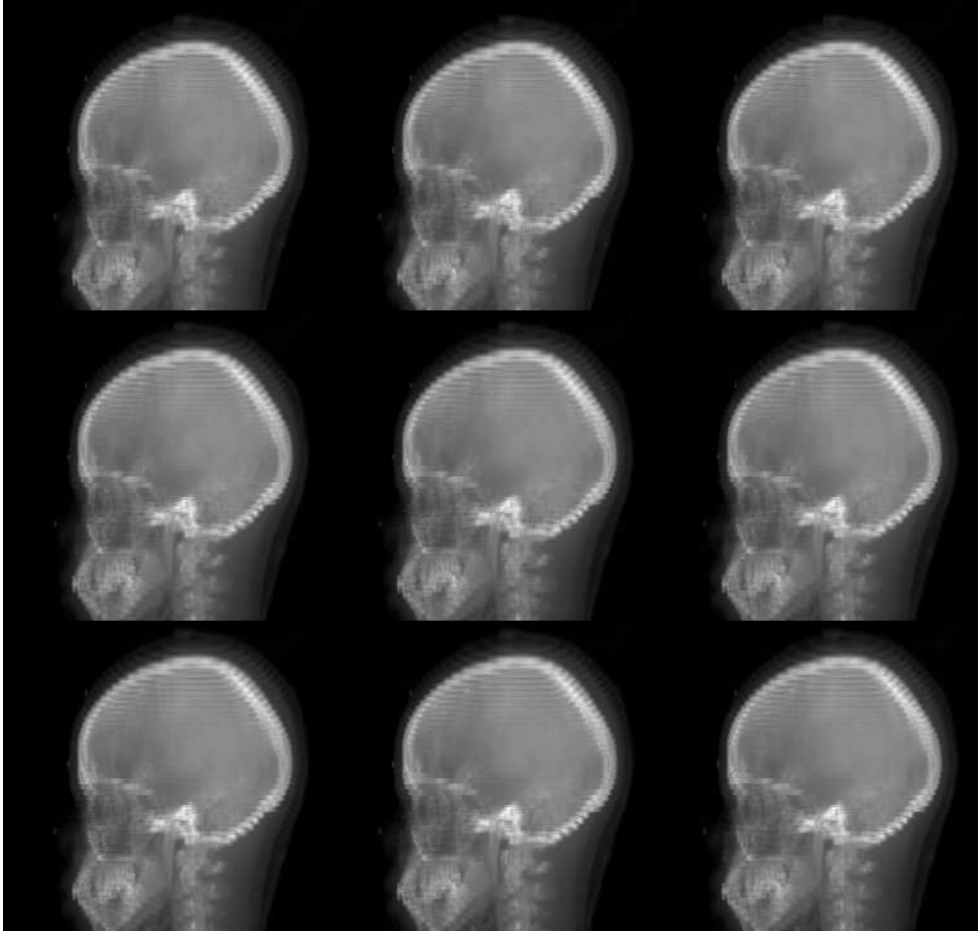


Figure 3.10: A light slab can be interpreted as a 2D collection of 2D images. Each image is taken from a different observation point. The (u, v) coordinates identify a sub-image in the light slab and the (s, t) coordinates the pixel in the sub-image.

3.2.1 Using light fields for DRR generation

A pixel value in a general light field is an indication of the amount of light reflected off the first surface a ray intersects with. When evaluating DRRs, however, the pixel values are the radiological path lengths (equation (2.2.1)) the rays encounter from the projection point to the image plane.

To accommodate the generation of DRRs, we can associate each point $\mathbf{p}_i = (u_i, v_i, s_i, t_i)$ with a scalar function $\mathbf{p}_i \mapsto q(\mathbf{p}_i)$ which maps a point to the radiological path length of the ray $R_{\mathbf{p}_i}$.

In order to trace a ray through the CT data and maintain the same parameterisation of rays in space as traditional light fields one must cast the rays beyond a *virtual image plane* onto an *effective image plane*. The values on the

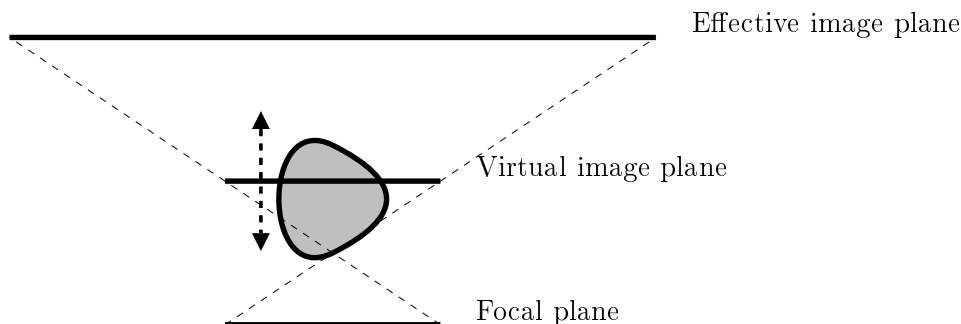


Figure 3.11: The positions of the focal-, effective image- and virtual image planes used when constructing DRRs using light fields. The object in grey is the CT data positioned relative to the planes. Note that the virtual image plane can be positioned anywhere between the focal- and effective image planes, as indicated by the arrows.

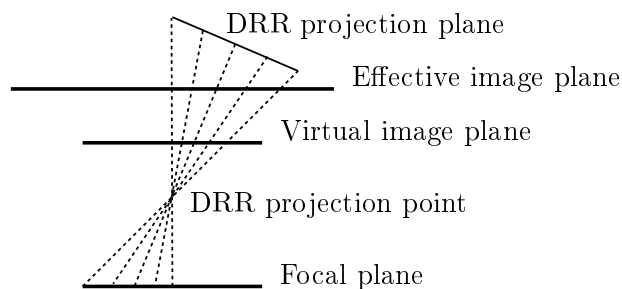


Figure 3.12: An example of how a DRR is constructed using a light slab. The dashed lines show the intersections of rays with the focal and virtual image planes. The effective image plane is shown for illustrative purposes only and is not used in the DRR construction.

effective image plane is used for the light field generation.

In traditional light field rendering as well as light field DRR generation, the generated image is a skewed perspective image. However, where in traditional light field rendering the image plane remains fixed and between the scene and the focal plane, in DRR generation the virtual image plane remains fixed while the effective image plane can move and the effective image plane lies on the other side of the scene from the focal plane. Figure 3.11 illustrates this. Figure 3.12 shows how an arbitrary DRR can be created from a light slab. For each ray from an arbitrary projection point to an arbitrary projection plane, the intersections with the focal and virtual image planes are computed. These intersections are then used to calculate the indices into the light slab as well as the weights used to perform the interpolation.

3.2.2 Compression

A large number of images is needed to effectively sample the 4D space. This can cause the light slab to be several gigabytes in size, which is more than most computers have available. A great deal of redundancy exists in the light slab data. There exist redundancy in s and t because of inter-pixel similarity and redundancy in u and v because of inter-image similarity (Russakoff *et al.*, 2003). These redundancies can be exploited using vector quantisation to compress the data. Vector quantisation is a lossy compression algorithm. A codebook of the most common vectors is created using a training set of data. A vector is then represented as an index into the codebook which points to a codeword that is closest to the vector. The quality of a codeword is determined by calculating the mean-squared error (MSE) between a source sample and the codeword sample (Levoy and Hanrahan, 1996).

The compression ratio r can be computed as

$$r = \frac{kb}{\log_2 N} \quad ,$$

where b is the number of bits per pixel, k is the number of pixels per codeword and N is the number of codewords.

As an example, consider a codebook consisting of $N = 2^{16} = 65536$ codewords. If each codeword represents a $2 \times 2 \times 2 \times 2$ tile of pixels in (u, v, s, t) space, then we have $k = 16$. If each pixel is represented as a floating point value 4-bytes in size (which is typically the case for DRRs), then a compression ratio of 1 : 4 will be achieved.

The codebook itself can also be compressed to increase the overall compression ratio, but this must be done with a lossless compression algorithm like Lempel-Ziv or Huffman coding (Levoy and Hanrahan, 1996).

3.3 Parallelisation of the ray cast and light field algorithms

Section §2.4 provided the theoretical background regarding parallelisation of algorithms. In this section we will discuss how parallelisation can be used to increase the performance of the ray cast and light field algorithms.

3.3.1 Trivial parallelisation

The ray cast and light field algorithms are both image-order algorithms. That is, the algorithm iterates over the pixels of the resulting DRR and computes the values one after the other. This is also the case when a light slab is constructed for use in the light field algorithm. Figure 3.13 shows how an image-order algorithm is executed. Since the computation of different pixel values in no way interfere with each other, the whole algorithm is a parallelisable task similar to

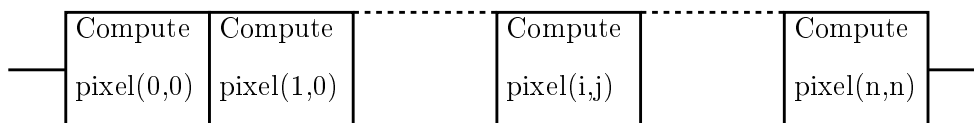


Figure 3.13: An image order algorithm. The whole algorithm is one big parallelisable task.

the ones in figure 2.15. This means that in terms of Amdahl's definition, the sequential fraction of the algorithm, F , is equal to zero and we can therefore expect a speedup of N times when N processors are used to parallelise the algorithm. We refer to this type of parallelisation as *trivial* as it requires very little overhead or modification to the algorithm to be implemented.

In contrast, object order algorithms are also parallelisable, but this would require locking mechanisms to prevent simultaneous updates of pixel values since multiple voxel projections affect the same pixels. Using locking mechanisms implies that certain parts of the code are serialised. Having serial code sections implies that F is not equal to zero. It should be clear that trying to parallelise object-order algorithms is more complex than parallelising image-order algorithms, it requires more overhead and the theoretical increase will not be as great as with image-order algorithms.

3.3.2 Adapting the algorithms

The parallel ray cast algorithm The adapted implementation:

```

#ifdef _OPENMP
#pragma omp parallel for
#endif
for (unsigned int j=0; j<yResolution; j++)
#ifdef _OPENMP
#pragma omp parallel for
#endif
    for (unsigned int i=0; i<xResolution; i++)
        image(i,j)=rayCast(focalPoint,
                           targetPoint(i,j),
                           volumeData);

```

The algorithm iterates over the pixel indices of the DRR image and for each pixel performs ray casting from a specified focal point to the target point associated with the pixel index. The target point is typically the centre of the pixel.

The parallel light field algorithm The adapted implementation:

```

#ifdef _OPENMP
#pragma omp parallel for
#endif
for (unsigned int j=0; j<yResolution; j++)
#ifdef _OPENMP
#pragma omp parallel for
#endif
    for (unsigned int i=0; i<xResolution; i++)
        image(i,j)=getLightSlabValue(focalPoint,
                                     targetPoint(i,j),
                                     lightSlabData);

```

The algorithm iterates over the pixel indices of the DRR image and for each pixel computes the value using a specified focal point and the target point associated with the pixel index as input the `getLightSlabValue` function. The `getLightSlabValue` function computes the intersections of the ray passing through the focal and target points with the focal and virtual image planes, as described in section §B.3. These intersections are then used to compute the interpolated value, as described in section §4.4.7.1.

The parallel light slab generation algorithm The adapted implementation:

```

#ifdef _OPENMP
#pragma omp parallel for
#endif
for (unsigned int u=0; u<uResolution; u++)
#ifdef _OPENMP
#pragma omp parallel for
#endif
    for (unsigned int v=0; v<vResolution; v++)
#ifdef _OPENMP
#pragma omp parallel for
#endif
        for (unsigned int s=0; s<sResolution; s++)
#ifdef _OPENMP
#pragma omp parallel for
#endif
            for (unsigned int t=0; t<tResolution; t++)
                Vector fp=focalPoint(u,v);
                Vector vp=virtualPoint(s,t);
                Vector tp=targetPoint(fp, vp);

```

```
lightSlabData(u,v,s,t)=rayCast(fp, tp,  
                               volumeData);
```

The algorithm iterates over the pixel indices of the focal and virtual image planes. For each iteration it computes a vector to a point on the focal plane and a point on the virtual image plane. These two points are used to compute the target point, which is the intersection of the ray passing through the focal plane point and virtual image plane point with the effective image plane. Ray casting is then performed from the focal point to the target point in order to compute the radiological path length which is stored in the light slab.

All three implementation examples use *nested parallel regions*, which will be discussed in more detail. A program using OpenMP starts out with a single initial thread of execution. When a thread encounters a parallel region, the thread creates a team consisting of itself as a master and zero or more other threads. All threads in the team execute the code in the parallel region, but only the master thread continues after it. When a thread encounters a nested parallel region the behaviour is similar. It checks if there are threads available to join the new team and the team executes the code in the parallel region.

On our hardware platform there are eight processors, which implies eight threads can be executed simultaneously. When the first parallel region is encountered, the first eight indices of the `for` loop are assigned to threads. When each of these encounters a nested parallel region there will be no other threads to join the team. The value of the nested constructs is only realised when the resolution of the image is not a multiple of the number of threads or when some threads take longer than others to complete, as would be the case when the ray cast algorithm is used. If the resolution of the image is not a multiple of eight, the last top level parallel region will eventually not use all eight threads and some of the second level teams that are created will contain more than one thread. The nested parallel regions ensure that all threads are used as soon as they have finished processing, since teams are repeatedly created at all levels. The creation of teams has no visible performance impact.

3.4 Summary

In this chapter the various techniques for improving the generation time of DRRs were discussed. Optimisation techniques for Siddon's algorithm were discussed and the use of light fields for DRR generation described. Parallelisation of the ray cast, light field and light slab generation algorithms was discussed in section §3.3. This chapter also demonstrated that these image-order algorithms are trivially parallelisable and that a performance increase equal to

the number of processors used can be expected when employing parallelisation techniques.

Chapter 4

Formulating the image registration problem

In this chapter the mathematical aspects of an image registration system and the DRR generation algorithms are formulated. Section §4.1 describes how the difference between two transformations are quantified. A total error and errors on the individual search space dimensions are defined. These are used when investigating the accuracy of the image registration system. Section §4.2 introduces qualitative and quantitative measures used to compare DRRs with each other and section §4.3 describes the optimiser used by the image registration implementation. Finally, section §4.4 formulates the computation of DRRs using the ray cast and light field algorithms and provides implementation examples.

4.1 Quantifying the error

In terms of our definitions

$$G_E = (R_z(\theta_z)R_y(\theta_y)R_x(\theta_x), (\delta_x, \delta_y, \delta_z)^T) \quad . \quad (4.1.1)$$

The components of equation (4.1.1) can also be represented in vector form:

$$\mathbf{v} = (\delta_x, \delta_y, \delta_z, \theta_x, \theta_y, \theta_z) \quad , \quad (4.1.2)$$

where \mathbf{v} is referred to as a position vector.

Let the *exact* (correct) solution to an optimisation process be called \mathbf{s}_e and the *calculated* solution \mathbf{s}_c , where both vectors are position vectors. The difference between the exact solution and the calculated solution is

$$\mathbf{s}_d = \mathbf{s}_e - \mathbf{s}_c \quad .$$

The first three elements of \mathbf{s}_d have *millimetre* units and the last three elements have *degree* units. In order to quantify the error between the two position

vectors we ideally need to get rid of the units. This is achieved by converting the differences of the individual elements to percentages. The size of the search space or allowed error range for a dimension is $2\epsilon_R$ if it has units in degrees and $2\epsilon_T$ if it has units in millimetres. We define a vector

$$\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5, \varepsilon_6) \quad ,$$

where

$$\varepsilon_i = \begin{cases} \frac{100 \times s_d^i}{2\epsilon_\delta} & \text{if } i = 1, \dots, 3 \\ \frac{100 \times s_d^i}{2\epsilon_\theta} & \text{if } i = 4, \dots, 6 \end{cases} \quad . \quad (4.1.3)$$

We can then calculate the Euclidean distance (L_2 norm) of $\boldsymbol{\varepsilon}$, which produces a dimensionless value quantifying the error. However, since the individual elements of $\boldsymbol{\varepsilon}$ have ranges of $[0 \dots 100]$, it follows that the total error has a range of $[0 \dots 100\sqrt{6}]$. For convenience we also scale this range to $[0 \dots 100]$. The definition of the total error quantification is

$$\xi(\mathbf{s}_d) = \frac{\|\boldsymbol{\varepsilon}\|_2}{\sqrt{6}} \quad . \quad (4.1.4)$$

Equation (4.1.4) provides the total error measured, but it is important to note that we are more interested in evaluating the errors on the individual elements as described in equation (4.1.3) when looking at the accuracy of the registration process.

4.2 Evaluation of DRR generation methods

There are various ways in which one can define the similarity of or difference between two images. Some, like a difference image, provide a qualitative representation that can be visually inspected, but are not suitable for use in algorithms. Quantitative measures exist, but usually these measures cannot be used to compare different sets of images.

4.2.1 Qualitative analysis of images

4.2.1.1 The absolute difference image

The absolute difference image d of two $M \times N$ images f and g is given by

$$d(m, n) = |f(m, n) - g(m, n)| \quad ,$$

where m and n are indices in the images.

Difference images are very useful in visualising the difference between two images, especially for synthetically created images. Unfortunately it is sometimes difficult to distinguish between relevant differences and noise when actual images are compared. Figure 4.1 shows a DRR, a rotated version of the same DRR and the difference image, with scaled values to show the difference more clearly.

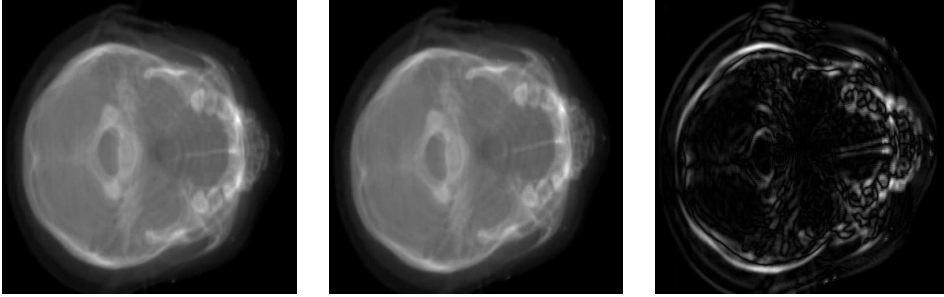


Figure 4.1: An example of a difference image. The first image shows a DRR, the second image is the same DRR rotated by 5° and the third image is the difference image scaled to show the differences more clearly.

4.2.2 Quantitative analysis of images

When evaluating the light field DRR generation method we have to show that the generated images have similarities that enable an optimiser to find the same optimum as when ray casted DRRs are used. Van der Bijl (2006) showed that the Mutual Information and Cross Correlation measures worked best with his optimiser.

4.2.2.1 Cross Correlation Coefficient

The cross correlation coefficient (or simply correlation coefficient) of two images f and g is defined as

$$CC(f, g) = \frac{\sum_{n=1}^N \sum_{m=1}^M (f(m, n) - \bar{f})(g(m, n) - \bar{g})}{\sqrt{\sum_{n=1}^N \sum_{m=1}^M (f(m, n) - \bar{f})^2 \sum_{n=1}^N \sum_{m=1}^M (g(m, n) - \bar{g})^2}} ,$$

where $f(m, n)$ and $g(m, n)$ denote the pixel values at position (m, n) in image f and g , respectively, and \bar{f} and \bar{g} denote the mean pixel value in image f and g , respectively.

4.2.2.2 Mutual Information

Entropy is a measure of information that stems from communication theory (Pluim *et al.*, 2003). In 1928 Hartley defined a measure of information for a message

$$H = n \log s = \log s^n , \quad (4.2.1)$$

where s is the number of possibilities per symbol in the message and n is the length of the message. Shannon noted that Hartley assumed that the likelihood of symbols occurring is equal. This will often not be the case, so Shannon adapted the measure to weigh the information of symbols by the

likelihood that the symbol will occur. Symbols that are less likely to occur contain more information. Shannon's entropy equation is

$$H = \sum_i p_i \log \frac{1}{p_i} = - \sum_i p_i \log p_i \quad , \quad (4.2.2)$$

with p_i the probability of symbol i occurring. Equation (4.2.2) reduces to equation (4.2.1) if the likelihood of all symbols occurring is equal. The entropy of an image is based on the distribution of the grey values of the image. A probability distribution can be calculated by counting the number of times each value occurs and dividing it by the total number of pixels. This is the same as calculating a histogram of the values, which is then scaled to form probabilities. The construction of the histogram requires that the image values are scaled and mapped to a discrete range.

Mutual Information was introduced as a registration measure in the early 1990's (Pluim *et al.*, 2003). It is useful to register multi-modal images, that is, images obtained from various sources such as CT, PET and MRI scans. The Mutual Information shared between two images f and g is defined as

$$MI(f, g) = H(f) + H(g) - H(f, g) \quad ,$$

where $H(f)$ and $H(g)$ are Shannon's entropies of images f and g , respectively, and $H(f, g)$ is the *joint* entropy of the two images.

To compute the entropies of DRR images, the values of the pixels, which are radiological path lengths, have to be mapped to a discrete range. The expectation probability of each value is then calculated. If the pixel values are mapped to the range [0..255], then the entropies are defined as

$$\begin{aligned} H(f) &= - \sum_{i=0}^{255} p_i \log p_i \\ H(f, g) &= - \sum_{i=0}^{255} \sum_{j=0}^{255} p_{ij} \log p_{ij} \quad , \end{aligned}$$

where p_i is the probability of pixel value i occurring and p_{ij} is the probability of pixel value i occurring in f given that pixel value j was observed in g .

4.3 The optimiser

Powell's method is an unbounded minimisation algorithm to determine local minima for multidimensional functions. It accomplishes the minimisation by repeatedly performing line minimisations. What makes Powell's method very attractive for the purposes of this thesis is the fact that it does not involve computation of the cost function's gradient (Greig, 1980). Firstly, no analytic

gradient exists for our cost function¹ and secondly, approximating the gradient numerically using finite difference or forward difference methods would be computationally expensive as it requires the generation of more DRRs.

Powell's method is part of a class of methods called *direction set methods*. All of these methods start out with a set of the unit vectors. It finds the minimum of the cost function by doing a line minimisation along the first unit vector, then the second and so forth. The simplest direction set method, called the Unit Vector Direction Set, repeatedly cycles through the set of unit vectors doing line minimisations until the cost function stops decreasing. For many problems UVDS performs well, but there are certain types of problems for which it performs badly.

Other direction set methods differ in the way which the set of unit vectors are updated as the method proceeds. Attempts are made to choose *conjugate* or "non-interfering" directions (described in section §4.3.2) with the property that minimisation along one of these directions is not undone by subsequent minimisation along another. This helps ensure that the algorithm converges.

The implementation of Powell's method used for the purposes of this thesis uses Brent's method to perform line minimisation. It is discussed in the following section.

4.3.1 Brent's method

A function f can be approximated by an $(N - 1)$ th order polynomial if N points are available. A quadratic polynomial is the simplest one that can have a minimum or maximum. It can be shown that given three points $(a, f(a))$, $(b, f(b))$ and $(c, f(c))$, the abscissa² is given by

$$x = b - \frac{1}{2} \frac{(b-a)^2 (f(b) - f(c)) - (b-c)^2 (f(b) - f(a))}{(b-a)(f(b) - f(c)) - (b-c)(f(b) - f(a))} . \quad (4.3.1)$$

The previous definition comes from Press *et al.* (1992), while Greig (1980) suggests the following form:

$$x = \frac{1}{2} \frac{(b^2 - c^2)f(a) + (c^2 - a^2)f(b) + (a^2 - b^2)f(c)}{(b-c)f(a) + (c-a)f(b) + (a-b)f(c)} . \quad (4.3.2)$$

Note that equations (4.3.1) and (4.3.2) only holds if the three points are not colinear, that is, not lying on the same line. If the points do lie on the same line, the denominator in both equations will be zero.

Brent's method uses the following variables:

a Lower bracket for the minimum.

b Upper bracket for the minimum.

¹Refer to section §4.3.4.

²The value of x where $f(x)$ is a minimum.

- x The point with the least value found so far. In case of a tie the newer value replaces the older value.
- w The point with the second least function value.
- v The previous value of w .
- u The point at which the function was evaluated most recently.

It attempts parabolic interpolation through points x , v and w . Since parabolic interpolation can jump to maxima as well as minima, the interpolation is only considered successful if both of the following holds:

1. The abscissa of the parabola falls within the interval (a, \dots, b) and
2. the abscissa of the parabola differs from the current best value x by less than half the movement of the step before the last one. This condition ensures that the algorithm converges.

If the parabolic steps are acceptable, but produces a maximum point, the method tries to alternate between using golden sections and parabolic steps. In such a case the use of the golden section method will ensure convergence. The reason why a comparison is done to the step before the last one is completely heuristic. Using the step before the last one prevents the algorithm from being unnecessarily negatively affected by a single bad step it can make up for in the next.

The algorithm typically ends when a and b are within a specified tolerance of each other, with the best abscissa $x = \frac{a+b}{2}$.

4.3.2 Conjugate directions

This section provides the mathematical definition of what was previously referred to as "non-interfering" or conjugate directions and comes from Press *et al.* (1992).

When minimising a function f along a direction \mathbf{u} , the gradient of the function will be perpendicular to \mathbf{u} at the minimum. If this was not the case then there would still be a derivative along \mathbf{u} that is not zero.

If we take a particular point \mathbf{p} as the origin of the coordinate system with coordinates \mathbf{x} , then any function f can be approximated by its Taylor series

$$\begin{aligned}
 f(\mathbf{x}) &= f(\mathbf{p}) + \sum_i \frac{\delta f(\mathbf{p})}{\delta x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\delta^2 f(\mathbf{p})}{\delta x_i \delta x_j} x_i x_j + \dots \\
 &\approx c - \mathbf{b}^T \cdot \mathbf{x} + \frac{1}{2} \mathbf{x}^T \cdot \mathbf{A} \cdot \mathbf{x} \quad , \quad (4.3.3)
 \end{aligned}$$

where

$$\begin{aligned} c &= f(\mathbf{p}) \\ \mathbf{b} &= -\nabla f|_{\mathbf{p}} \\ A_{ij} &= \frac{\delta^2 f}{\delta x_i \delta x_j}|_{\mathbf{p}} \quad . \end{aligned}$$

The matrix \mathbf{A} is called the *Hessian matrix* of function f at \mathbf{p} . It has the second partial derivatives of the function f as components.

In the Taylor approximation equation (4.3.3) the gradient of f is calculated as

$$\nabla f = \mathbf{A} \cdot \mathbf{x} - \mathbf{b} \quad . \quad (4.3.4)$$

We know that the gradient at any extremum is zero, so we can obtain these extrema by solving for \mathbf{x} in the equation $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$.

The change in the gradient ∇f is given by

$$\delta(\nabla f) = \mathbf{A} \cdot (\delta\mathbf{x}) \quad . \quad (4.3.5)$$

If we have moved to a minimum along a direction \mathbf{u} and are trying to determine a new direction \mathbf{v} that will not spoil our minimisation along \mathbf{u} , we have to choose \mathbf{v} so that it is perpendicular to \mathbf{u} . This is the same as requiring the change in the gradient to be perpendicular to \mathbf{u} . Using equation (4.3.5) this gives us

$$0 = \mathbf{u} \cdot \delta(\nabla f) = \mathbf{u} \cdot \mathbf{A} \cdot \mathbf{v} \quad . \quad (4.3.6)$$

The two vectors \mathbf{u} and \mathbf{v} are said to be *conjugate* when equation (4.3.6) holds. When the relation holds pairwise for all members of a vector set, it is called a conjugate set.

When doing successive line minimisations of a function using a conjugate set of directions there is no need to redo any of those directions. If a direction set method can construct N linearly independent, mutually conjugate directions then one pass of N line minimisations will put it exactly at the minimum of a quadratic form like equation (4.3.3). Functions that are not exact quadratic forms won't be exactly at the minimum, but repeated cycles of N line minimisations will converge *quadratically* to the minimum.

4.3.3 Powell's method

Powell discovered a direction set method that produces N mutually conjugate directions. The following description comes from Press *et al.* (1992). Greig (1980) also provides a description.

Initialise the set of directions \mathbf{u}_i to the basis vectors,

$$\mathbf{u}_i = \mathbf{e}_i \quad i = 1, \dots, N \quad .$$

The following steps are then repeated:

1. Save the starting position, \mathbf{p}_0 .
2. Perform a line minimisation along the line passing through the point \mathbf{p}_{i-1} with direction \mathbf{u}_i and call the minimum point \mathbf{p}_i . This is done for $i = 1, \dots, N$.
3. Update the direction set with $\mathbf{u}_i \leftarrow \mathbf{u}_{i+1}$, where $i = 1, \dots, N - 1$.
4. Add a new direction vector to the direction set: $\mathbf{u}_N \leftarrow \mathbf{p}_N - \mathbf{p}_0$.
5. Perform a line minimisation along the line passing through point \mathbf{p}_N with direction \mathbf{u}_N and set \mathbf{p}_0 equal to this minimum. Repeat from step 2.

The choice of line minimisation algorithm is arbitrary and can be described in general as an algorithm that, given a starting point \mathbf{p} , a direction \mathbf{u} and a function f , finds the scalar λ that minimises $f(\mathbf{p} + \lambda\mathbf{u})$. The implementation used in this thesis uses Brent's algorithm (section §4.3.1) to perform the line minimisations.

The abovementioned procedure has the drawback that repeated usage of $\mathbf{p}_N - \mathbf{p}_0$ to construct new directions may lead to direction sets that are linearly dependent. This will lead the optimiser to only search a subspace of the N -dimensional space and produce an incorrect answer. Various methods, ranging from resetting the direction set to heuristic approaches, exist to address this problem. This thesis uses the implementation of Powell's algorithm provided by the VXL software package, which makes use of a heuristic approach.

4.3.4 The cost function

The cost function is defined as a minimising function. This function needs to be defined for all $\mathbf{x} \in \mathbb{R}^6$, since Brent's minimiser and by implication also Powell's minimiser are unconstrained optimisers. Using ray casting we can easily generate arbitrary DRRs, but using the light field algorithm we are constrained to the sampled space contained in the light slab. To overcome this limitation we define an arbitrary function with the criteria that it

- provides values for points outside the sampled space and
- guides the optimiser to the minimum.

Let $g(\mathbf{x})$ be a similarity measure defined in the interval $-\epsilon \leq x_i \leq \epsilon$ for all six dimensions. It compares two DRRs and returns lower values for higher similarity and higher values for lower similarities.

Let $c(\mathbf{x})$ be a cost function, then

$$c(\mathbf{x}) = \begin{cases} -x_i & \text{if } x_i < -\epsilon \\ g(\mathbf{x}) & \text{if } -\epsilon \leq x_i \leq \epsilon \\ x_i & \text{if } \epsilon < x_i \end{cases}$$

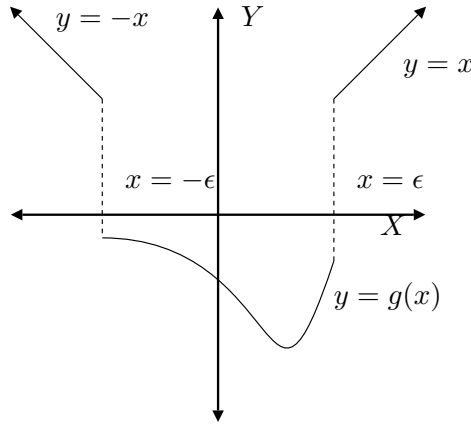


Figure 4.2: A 2D illustration of the unconstrained cost function $c(x)$.

is a function that is unconstrained and guides the optimiser to the minimum. Figure 4.2 provides a graphical representation of a 2D view of the unconstrained cost function. Note that the function $c(\mathbf{x})$ contains discontinuities at $-\epsilon$ and ϵ in all six dimensions.

The two functions investigated in this thesis as possibilities for $g(\mathbf{x})$ are the Mutual Information (section §4.2.2.2) and Correlation Coefficient (section §4.2.2.1) similarity measures. Note that when references are made to the "cost function", it always refers to the unconstrained version $c(\mathbf{x})$. Also, since the Mutual Information and Cross Correlation similarity measures both always return positive results, it must be negated when used in the cost function. So, for Mutual Information,

$$g(\mathbf{x}) = -MI(DRR(\mathbf{x}), DRR_{ref}) \quad ,$$

where $DRR(\mathbf{x})$ is a function returning the DRR when a transformation \mathbf{x} is used and DRR_{ref} is a reference image. The same applies when using the Correlation Coefficient similarity measure.

4.4 Formulating the DRR generation problem

4.4.1 Coordinate systems

The CT coordinate system \mathcal{C} The CT coordinate system is a right-handed coordinate system. It is used to represent the data obtained during the CT scan and is also the coordinate system which the ray casting algorithm uses when constructing a DRR.

The patient coordinate system \mathcal{P} The patient coordinate system is a right-handed coordinate system that differs from the CT coordinate system \mathcal{C} by a fixed transformation G_{PC} .

The beam coordinate system \mathcal{B} The beam coordinate system is a right-handed coordinate system with its origin at the treatment isocentre in the treatment vault. The x -axis coincides with the beam axis, with the positive x -axis directed in the beam direction. The positive z -axis is directed vertically upwards and the y -axis lies in the horizontal plane. The treatment plan defines the transformation G_{BP} , which will position the patient correctly in the beam line.

The detector coordinate system \mathcal{D} The detector is a rectangular device used to pick up X-ray values which it then digitises. The detector coordinate system \mathcal{D} is a right handed-coordinate system with its origin located in the centre of the detector. The positive y -axis is directed upwards and is parallel to the left and right sides of the detector. The positive x -axis is parallel to the top and bottom sides of the detector and on the right side of the detector when viewed from the front. The positive z -axis is orthogonal to the x - and y -axis. The transformation from the detector coordinate system to the beam coordinate system is defined as G_{BD} and determined during calibration of the system.

The light field coordinate system \mathcal{L} The light field coordinate system is a synthetic right-handed coordinate system. By construction the origin of this coordinate system coincides with the origin of the beam coordinate system. The negative x -axis is chosen so that the X-ray source point lies on it. The y -axis is orthogonal to the x -axis of the light field coordinate system and the y -axis of the detector coordinate system. Lastly, the z -axis of the light field coordinate system is orthogonal to the x - and y axis of the light field coordinate system. The two planes used to construct the light slab, the focal and virtual image planes, are assumed to be rectangular and parallel. The the distance separating the planes is f . The virtual image plane lies in the yz coordinate plane with its centre at the origin. The focal plane is f units away at the X-ray source on the negative x -axis.

4.4.2 Vectors and points used to construct DRRs

\mathbf{p}_P^t The target point (centre of the tumour) defined in the patient coordinate system.

\mathbf{p}_P^e The entry point of the beam on the patient's anatomy defined in the patient coordinate system.

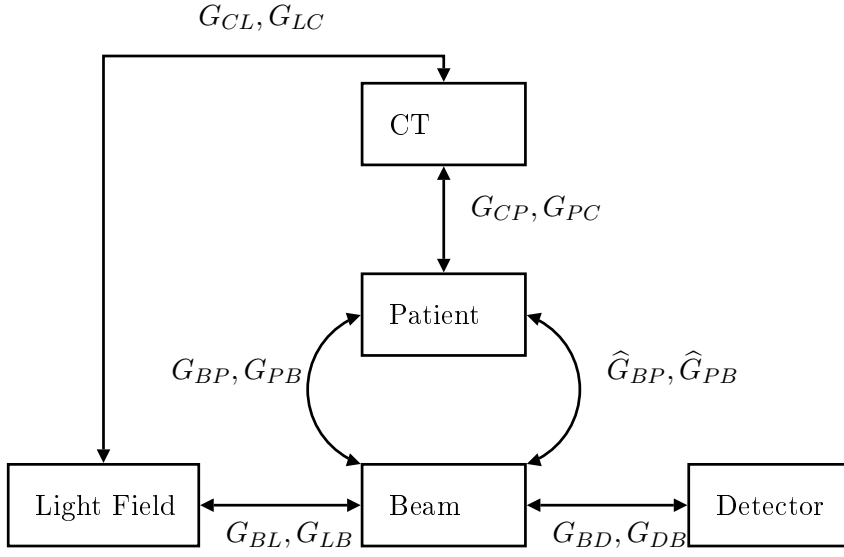


Figure 4.3: The relationships between the various coordinate systems. \hat{G}_{PB} and \hat{G}_{BP} are transformations calculated by the SPG.

μ_P A unit vector orthogonal to the vector $\mathbf{v}_P = \mathbf{p}_P^t - \mathbf{p}_P^e$. It corresponds to the upward direction when looking at the patient through the beam's eye view of the planning system and is used to calculate the collimator rotation, for example.

\mathbf{p}_B^s The X-ray source point defined in the beam coordinate system.

4.4.3 Transformations

G_{PC} A fixed relationship between the CT and patient coordinate systems.

G_{BP}^0 A transformation from the patient to the beam coordinate system which has the property that when it is applied to \mathbf{p}_P^t , \mathbf{p}_P^e and μ_P , the following holds:

1. $\mathbf{p}_B^t - \mathbf{p}_B^e$ coincides with and has the same direction as the positive x axis of the beam coordinate system
2. $\mathbf{p}_B^t = \mathbf{0}_B$, the origin of the beam coordinate system
3. μ_B coincides with and has the same direction as the positive z axis of the beam coordinate system

This transformation is needed since a relationship between the patient and beam coordinate system is required pre-operatively. The detector coordinates, defined in the beam coordinate system, are used to determine the range of the sampling space.

G_{BP}, θ A transformation from the patient to the beam coordinate system defined in the same way as G_{BP}^0 , with the exception that no restriction is placed on the orientation of $\boldsymbol{\mu}_B$. The angle θ is calculated from the orientation of $\boldsymbol{\mu}_B$. The following equations always hold:

$$\boldsymbol{\mu}_B = G_{BP} \diamond \boldsymbol{\mu}_P \quad (4.4.1)$$

$$\theta = \text{atan2}(\mu_B^y, \mu_B^z) \quad (4.4.2)$$

$$G_{BP}^0 = (R_x^T(\theta), \mathbf{0}) \diamond G_{BP} \quad (4.4.3)$$

The function $\text{atan2}(x, y)$ calculates the arc tangent of the two variables x and y . It is similar to calculating the arc tangent of $\frac{y}{x}$, except that the signs of both arguments are used to determine the quadrant of the result.

G_E A transformation that describes the error in the positioning of the patient. This transformation is constrained to a small space, with limits on the rotation and translation parameters. This error transformation includes both errors from fitting the mask as well as possible errors from the SPG. This transformation is defined by equation (4.1.1) and can also be represented as a six dimensional vector as in equation (4.1.2).

\widehat{G}_{BP} A transformation from the patient to the beam coordinate system calculated by the SPG system. This transformation may contain errors. The following equation always holds:

$$G_{BP} = G_E^{-1} \diamond \widehat{G}_{BP} \quad (4.4.4)$$

Here G_E^{-1} , the inverse of G_E , is interpreted as the transformation that *corrects* any positioning errors.

G_{LB}, G_{LP} Two transformations defining the transformation to the light slab coordinate system. Theoretically only G_{LP} is needed, but from a practical point it is convenient to use G_{LB} . The two transformations are related in such a way that

$$G_{LB} = G_{LP} \diamond G_{PB}^0 \quad (4.4.5)$$

Section §4.4.5 describes how this transformation is constructed.

G_{BD} A fixed transformation between the detector and beam coordinate systems obtained during a calibration process.

4.4.4 Creating a DRR using conventional ray casting

To create a DRR using the ray casting algorithm, one simply needs to compute the X-ray source point and detector sampling points in the CT coordinate

system. Ray casting can then be performed to calculate the pixel values. Converting coordinates can be done using the theoretical transformation G_{BP} or the transformation \widehat{G}_{BP} calculated by the SPG, depending on the situation. For example, the X-ray source point \mathbf{p}^s in the CT coordinate system can be calculated by the following two equations:

$$\begin{aligned}\mathbf{p}_C^s &= G_{CP} \diamond G_{PB}^0 \diamond \mathbf{p}_B^s \\ \mathbf{p}_C^s &= G_{CP} \diamond \widehat{G}_{PB} \diamond \mathbf{p}_B^s \quad .\end{aligned}$$

4.4.5 Determining the transformations G_{BL} and G_{LB}

The transformation G_{BL} is defined as the transformation from the light field coordinate system to the beam coordinate system. It is used during the construction of the light slab. From our description of the construction of the light field coordinate system in section §4.4.1 it follows that the following must hold:

$$R_{BL}\mathbf{i}_L = \mathbf{u}_B = \frac{\mathbf{0}_B - \mathbf{p}_B^s}{\|\mathbf{0}_B - \mathbf{p}_B^s\|_2} \quad (4.4.6)$$

$$R_{BL}\mathbf{j}_L = \mathbf{v}_B = \mathbf{k}_B \times (G_{BD} \diamond \mathbf{j}_D) \quad (4.4.7)$$

$$R_{BL}\mathbf{k}_L = \mathbf{w}_B = \mathbf{u}_B \times \mathbf{v}_B \quad (4.4.8)$$

$$G_{BL} \diamond \mathbf{0}_L = G_{BD} \diamond \mathbf{0}_B \quad . \quad (4.4.9)$$

From equation (4.4.9) it follows directly that

$$\mathbf{t}_{BL} = \mathbf{t}_{BD}$$

and from equations (4.4.6), (4.4.7) and (4.4.8) it follows that

$$R_{BL} = [\mathbf{u}_B \ \mathbf{v}_B \ \mathbf{w}_B] \quad .$$

The transformation G_{LB} is defined as the transformation from the beam coordinate system to the light field coordinate system. It is used during the construction of a DRR from the light slab. The definition of G_{LB} is obtained directly from the definition of G_{BL} :

$$G_{LB} = (R_{LB}, \mathbf{t}_{LB}) = (R_{BL}^{-1}, -R_{BL}^{-1}\mathbf{t}_{BL}) \quad .$$

R_{BL} is a rotation matrix and thus always invertible.

4.4.6 Constructing the light slab

In order to construct a light slab, one needs to know

- the transformation from the light field coordinate system to the CT coordinate system $G_{CL} = G_{CP} \diamond G_{PB}^0 \diamond G_{BL}$ and

- the sampling densities of the focal and virtual image planes.

The following steps describe the construction of the light slab:

1. Determine the offsets and lengths of the focal and virtual image planes.
2. Allocate the necessary memory using the sampling densities to calculate the required amount.
3. Convert light slab coordinates to CT coordinates and perform ray casting to calculate the light slab values.

4.4.6.1 Determining the offsets and lengths of the focal and virtual image planes

When constructing the light slab, the first step is to determine the offset of the virtual image and focal planes. For our purposes the virtual image plane is chosen to contain the target point \mathbf{p}_B^t and is orthogonal to the vector $\mathbf{p}_B^t - \mathbf{p}_B^e$. The focal plane is parallel to the virtual image plane with the same distance between the planes as the distance between the isocentre and the X-ray source in the beam coordinate system. When determining the offsets of the planes we need to make provision for all possible values of \widehat{G}_{PB} , that is, for any point \mathbf{p}_B in the beam coordinate system, the corresponding point

$$\mathbf{p}_L = G_{LP} \diamond \widehat{G}_{PB} \diamond \mathbf{p}_B \quad (4.4.10)$$

must be sampled. Since we cannot get \widehat{G}_{PB} from the SPG pre-operatively, we must expand equation (4.4.10) using equations (4.4.4) and (4.4.3) to

$$\begin{aligned} \mathbf{p}_L &= G_{LP} \diamond \widehat{G}_{PB} \diamond \mathbf{p}_B \\ &= G_{LP} \diamond G_{PB} \diamond G_E^{-1} \diamond \mathbf{p}_B \\ &= G_{LP} \diamond G_{PB}^0 \diamond (R_x^T(\theta), \mathbf{0}) \diamond G_E^{-1} \diamond \mathbf{p}_B \quad . \end{aligned} \quad (4.4.11)$$

As mentioned earlier, it is easier to use G_{LB} in practice. When using G_{LB} equation (4.4.11) reduces to

$$\begin{aligned} \mathbf{p}_L &= G_{LP} \diamond G_{PB}^0 \diamond (R_x^T(\theta), \mathbf{0}) \diamond G_E^{-1} \diamond \mathbf{p}_B \\ &= G_{LB} \diamond (R_x^T(\theta), \mathbf{0}) \diamond G_E^{-1} \diamond \mathbf{p}_B \quad . \end{aligned} \quad (4.4.12)$$

A brute force search is performed pre-operatively using the four detector points, the X-ray source point and varying values for G_E and θ to determine the minimum and maximum intersection coordinates with the planes. The minimum coordinates are used as the offsets (\mathbf{c}_{uv}^L and \mathbf{c}_{st}^L) of the focal and virtual image planes and the minimum and maximum coordinates are used to compute the lengths (\mathbf{l}_{uv} and \mathbf{l}_{st}) of the focal and virtual image planes.

Russakoff, Rohlfing, Mori, Rueckert, Ho, Adler & Maurer (2005) proposes an exact geometrical solution to determine the focal and virtual image plane

sizes and offsets. A simpler alternative computation which approximates the answer is also suggested in their paper. Their solution was *not* used as it made the following assumptions:

1. The X-ray detector is square,
2. the X-ray source point \mathbf{p}_B^s lies in the beam line, that is on the x -axis of the beam coordinate system,
3. the centre of the detector lies on the beam line as well and
4. the detector is orthogonal to the beam line.

Assumptions (2), (3) and (4) are exactly what would be aimed for when physically constructing the system, but is not easily attainable in practice.

4.4.6.2 Convert light slab coordinates to CT coordinates and perform ray casting to calculate the light slab values

When the offsets and lengths of the planes have been computed, the sampling is done by iterating over all pairs of sampling coordinates on both planes. The transformation

$$G_{LC} = G_{LB} \diamond G_{BP}^0 \diamond G_{PC}$$

is used to map points from the light field coordinate system to the CT coordinate system.

Instead of performing the transformation G_{CL} on each sampling point on the focal and virtual image planes, it is more efficient to transform only 3 points on each plane: one to act as a reference point or offset and the two others to determine the deltas between sampling points in the u and v (or similarly s and t) directions. We define the offsets of the focal and virtual image planes in the CT coordinate system as

$$\begin{aligned} \mathbf{c}_{uv}^C &= G_{CL} \diamond \mathbf{c}_{uv}^L \\ \mathbf{c}_{st}^C &= G_{CL} \diamond \mathbf{c}_{st}^L \quad . \end{aligned}$$

The increments between sampling points on the focal and virtual image planes in the CT coordinate system are given by

$$\begin{aligned} \Delta_u^C &= R_{CL}(0, \frac{l_{uv}^y}{s_u}, 0)^T \\ \Delta_v^C &= R_{CL}(0, 0, \frac{l_{uv}^z}{s_v})^T \\ \Delta_s^C &= R_{CL}(0, \frac{l_{st}^y}{s_s}, 0)^T \\ \Delta_t^C &= R_{CL}(0, 0, \frac{l_{st}^z}{s_t})^T \quad , \end{aligned}$$

where s_u , s_v , s_s and s_t are the number of sampling intervals³ for the respective dimensions.

The calculation of the light slab then reduces to the following algorithm:

```

for (u=0; u <= su; u++)
  for (v=0; v <= sv; v++)
    for (s=0; s <= ss; s++)
      for (t=0; t <= st; t++)
        focalPoint ← cuvC + uΔuC + vΔvC
        virtualPoint ← cstC + sΔsC + tΔtC
        targetPoint ← intersect(focalPoint, virtualPoint,
                               effectiveImagePlane)
        point(u, v, s, t) ← raycast(focalPoint, targetPoint,
                                    volumeData)

```

The ray must be extended beyond the virtual image plane onto an effective image plane. This plane must never intersect with the CT data, so it is defined to lie on the opposite side of the detector on the x axis than where the patient is positioned. The focal and virtual image plane points are used to compute the target point. The target point is the intersection of the ray passing through the focal plane point and virtual image plane point with the effective image plane. Section §B.3 describes how to calculate the intersection of a line with a plane.

4.4.7 Creating a DRR from a light slab

During treatment the SPG supplies the transformation from the patient coordinate system to the beam coordinate system, \widehat{G}_{BP} . The image registration system tries to find a matching DRR in a search space limited by G_E . For a G_E chosen by the optimiser, equations (4.4.11) and (4.4.12) still holds, but it is important to note that here θ (equation (4.4.2)) is computed using

$$\boldsymbol{\mu}_B = G_E^{-1} \diamond \widehat{G}_{BP} \diamond \boldsymbol{\mu}_P \quad , \quad (4.4.13)$$

which is derived from equations (4.4.1) and (4.4.3).

To create a DRR using the light slab involves the following steps:

1. Compute the X-ray source point using the optimiser estimation G_E and equations (4.4.12), (4.4.13) and (4.4.2):

$$\boldsymbol{p}_L^s = G_{LB} \diamond (R_x^T(\theta), \mathbf{0}) \diamond G_E^{-1} \diamond \boldsymbol{p}_B^s \quad .$$

³Note that the number of sampling intervals are exactly one less than the number of samples for each dimension.

2. Compute the sampling points on the detector in the light field coordinate system also using the optimiser estimation:

$$\mathbf{d}_L = G_{LB} \diamond (R_x^T(\theta), \mathbf{0}) \diamond G_E^{-1} \diamond G_{BD} \diamond \mathbf{d}_D \quad .$$

3. For each ray passing through the X-ray source point and a sampling point, find the intersections with the focal and virtual image planes as described in section §B.3. Perform quadrilinear interpolation using the computed intersections as described in section §4.4.7.1. to find the required pixel value.

This was illustrated in figure 3.12.

4.4.7.1 Quadrilinear interpolation

In linear interpolation the value of a point u is dependent on its nearest neighbours in such a way that $u = \alpha_0 u_0 + \alpha_1 u_1$, where

$$\begin{aligned} \alpha_0 &= \frac{u_1 - u}{u_1 - u_0} \\ \alpha_1 &= \frac{u - u_0}{u_1 - u_0} = 1 - \alpha_0 \quad . \end{aligned}$$

Similarly, for quadrilinear interpolation the value of a point $p(u, v, s, t)$ is given by the formula

$$p(u, v, s, t) = \sum_{i=0}^1 \sum_{j=0}^1 \sum_{k=0}^1 \sum_{l=0}^1 \alpha_i \beta_j \gamma_k \delta_l p(u_i, v_i, s_i, t_i) \quad .$$

Let \mathbf{d}_L denote an arbitrary detector sampling point in the light field coordinate system. The value associated with a ray passing through the X-ray source point \mathbf{p}_L^s and the arbitrary detector sampling point \mathbf{d}_L is computed by first finding the intersection points of the same ray with the focal and the effective image planes. Let \mathbf{a}_{uv} and \mathbf{a}_{st} denote these intersections. The next step is to determine the sample indices on the focal and virtual image planes that will be used. We subtract the plane offsets from the plane intersections and then divide by the number of sampling intervals to find the first index for

each axis on each plane. The following then holds:

$$\begin{aligned}
u_0 &= \left\lfloor \frac{(\mathbf{a}_{uv} - \mathbf{c}_{uv})^T \mathbf{j}}{\Delta_u} \right\rfloor \\
v_0 &= \left\lfloor \frac{(\mathbf{a}_{uv} - \mathbf{c}_{uv})^T \mathbf{k}}{\Delta_v} \right\rfloor \\
s_0 &= \left\lfloor \frac{(\mathbf{a}_{st} - \mathbf{c}_{st})^T \mathbf{j}}{\Delta_s} \right\rfloor \\
t_0 &= \left\lfloor \frac{(\mathbf{a}_{st} - \mathbf{c}_{st})^T \mathbf{k}}{\Delta_t} \right\rfloor \\
u_1 &= u_0 + 1 \\
v_1 &= v_0 + 1 \\
s_1 &= s_0 + 1 \\
t_1 &= t_0 + 1 \\
u &= (\mathbf{a}_{uv} - \mathbf{c}_{uv})^T \mathbf{j} \pmod{\Delta_u} \\
v &= (\mathbf{a}_{uv} - \mathbf{c}_{uv})^T \mathbf{k} \pmod{\Delta_v} \\
s &= (\mathbf{a}_{st} - \mathbf{c}_{st})^T \mathbf{j} \pmod{\Delta_s} \\
t &= (\mathbf{a}_{st} - \mathbf{c}_{st})^T \mathbf{k} \pmod{\Delta_t} \quad .
\end{aligned}$$

4.5 Summary

This chapter formulated the problem of computing DRRs using the ray cast and light field algorithms for use in the proton treatment facility at iThemba LABS. A formal definition of the measure of an error in the registration process was given. This was followed by an explanation of the different measures of comparing DRRs for similarity, which included the absolute difference image for visual comparison and the Mutual Information and Correlation Coefficient similarity measures for use in the image registration algorithm. Powell's optimiser and the definition of the cost function were discussed and finally the DRR generation problem was formulated.

Chapter 5

Experiments and results

In this chapter a number of tests are conducted to determine whether an image registration system using the light field algorithm to generate DRRs is usable in practice. In particular we wish to determine whether such a system is robust, that is it does not find incorrect solutions, accurate and fast enough to shorten the verification process significantly.

Van der Bijl (2006) showed that his proposed image registration system worked well with ray casted DRRs. In section §5.1 the relationship between the similarity curves of ray casted DRRs and DRRs generated from a light slab is investigated. As will be seen from the results of these measurements, the equivalence of the similarity curves is dependent on the light slab sampling densities and for most cases they correspond to the ray cast similarity curves.

Constructing a light slab is computationally expensive. Although the construction of the light slab is done pre-operatively and therefore does not add to the patient position verification time, it is worth noting the light slab generation times. Section §5.2 details the results of the light slab generation time experiments, whilst section §5.3 displays the time it took to create individual DRRs. The times of the ray cast and light field algorithms in serial and parallelised forms are listed and shows a dramatic performance increase when using the parallelised light field algorithm as opposed to the serial ray cast algorithm.

Finally, the parallelised light field algorithm is used with an optimiser to perform image registration. The accuracy of the optimiser's results is listed in section §5.4 along with information on the number of cost function evaluations, the number of DRRs generated and the total time required to perform the optimisation.

From the results in this section it is clear that the image registration process is fast and as accurate as when ray casted DRRs are used and therefore fulfils two of the major aims set out in this thesis. None of the arbitrary optimiser tests gave incorrect results. Furthermore, the equivalence of the ray cast and light slab similarity curves gives a good indication that an optimiser using

DRRs created by these two methods will behave similarly. This shows that the third aim, robustness, has also been achieved.

Note that none of the tests use images from the digital X-ray detector as it was not operational when this thesis was completed. The ray cast algorithm is used to generate reference DRRs.

5.1 Similarity tests

We construct a similarity experiment in order to evaluate the effect of using DRRs from various light slab configurations on the cost function of the optimiser. The effect is compared relative to the similarity performance of ray casted DRRs using the incremental techniques described in section §3.1.1.

Van der Bijl (2006) showed that, when determining the similarity measures between a reference DRR and some ray casted DRRs in its vicinity, the similarity measures peaked where the DRRs matched and gradually worsened in a decreasing fashion as a DRR is positioned further away from the reference DRR. For the light field DRRs to be useful, we expect that

- the similarity curves are smooth, or in other words, it does not contain local minima or maxima (extrema) and
- the location of the extremum must coincide with the extremum of the ray casted DRRs.

One might consider commenting on the gradient of the curves. Insisting that the gradient of a light field DRR similarity curve must not be much worse than that of the ray cast DRR curve seems natural, but at this stage we cannot yet determine the acceptable gradient range as it will depend on the generation time of the light field DRRs and the type of optimiser. We can at most theorise that a relative decline in the gradient must at least be complemented by a relative increase in DRR generation time in order to prevent the optimiser from performing slower when using light field DRRs than when using ray cast DRRs.

The similarity measurements are taken for movements along the six error dimensions as well as four other arbitrary complex movements. The different error dimensions are all limited to $\pm 5\text{mm}$ or $\pm 5^\circ$. On the graphs all movements are parameterised to the interval $[0 \dots 1]$ and are expected to peak at $\frac{1}{2}$. Each test is sampled at 51 evenly spaced intervals. The light slab configurations are varied as follows:

- the *focal plane* resolution varies between 8×8 and 64×64
- the *image plane* resolution varies between 128×128 and 512×512

Note that the combination of an image plane resolution of 512×512 and a focal plane resolution of 64×64 is not included in the tests as the memory requirement made it infeasible.

The light field DRR similarity curves relative to the ray casted DRR similarity curves are shown in figures 5.1 to 5.10. The solid red line in all the graphs is the similarity curve of ray casted DRRs, which is used as a reference. The test was performed on four different positions of the CT cube. Since there are no significant differences between the results of these four tests, only one result set is presented. The Mutual Information graphs were plotted using the range $[0.4 \dots 2.2]$ and the Correlation Coefficient graphs have a range of $[0.86 \dots 1.02]$.












Colour	Line		Focal plane	Image plane
Green	Solid		64×64	256×256
Blue	Solid		64×64	128×128
Magenta	Solid		32×32	512×512
Cyan	Solid		32×32	256×256
Red	Dotted		32×32	128×128
Green	Dotted		16×16	512×512
Blue	Dotted		16×16	256×256
Magenta	Dotted		16×16	128×128
Cyan	Dotted		8×8	512×512
Red	Dot-dashed		8×8	256×256
Green	Dot-dashed		8×8	128×128

Table 5.1: The legend to the similarity test graphs. The solid red line indicates the reference curve generated using the ray casted DRRs. The rest of the curves are generated from light slabs.

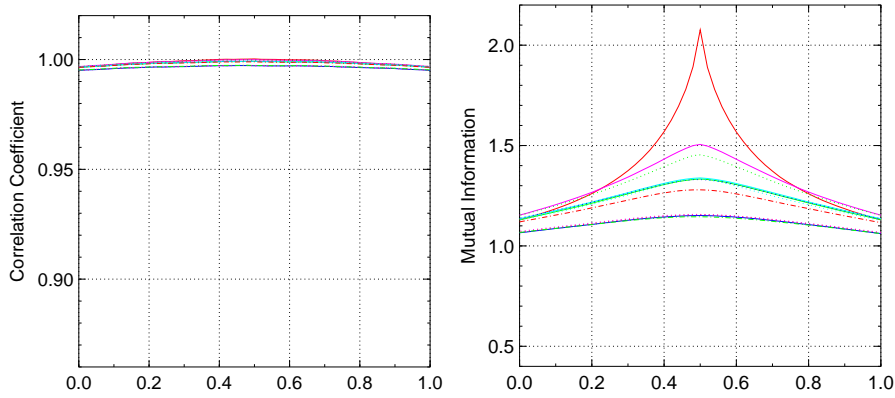


Figure 5.1: Similarity measure for translations along the x axis

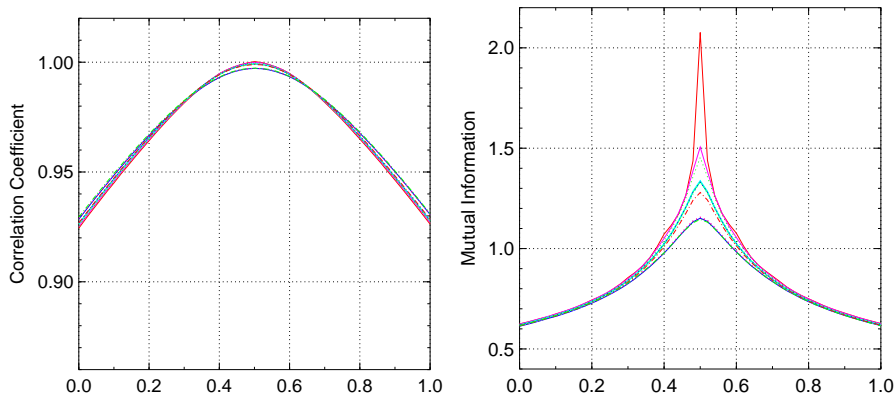


Figure 5.2: Similarity measure for translations along the y axis

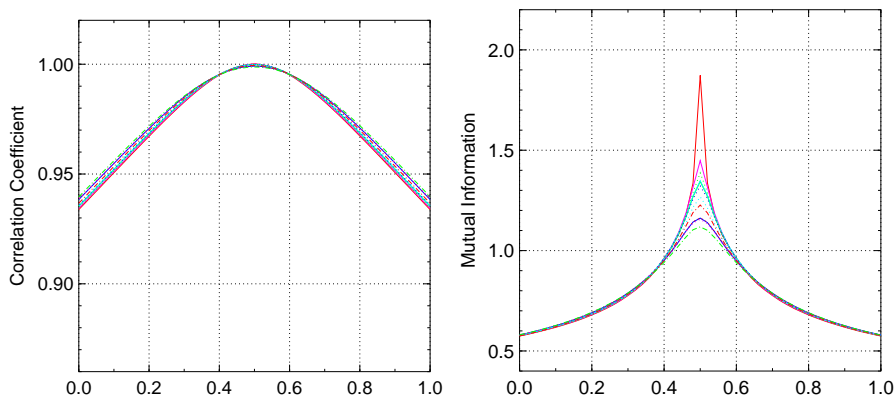


Figure 5.3: Similarity measure for translations along the z axis

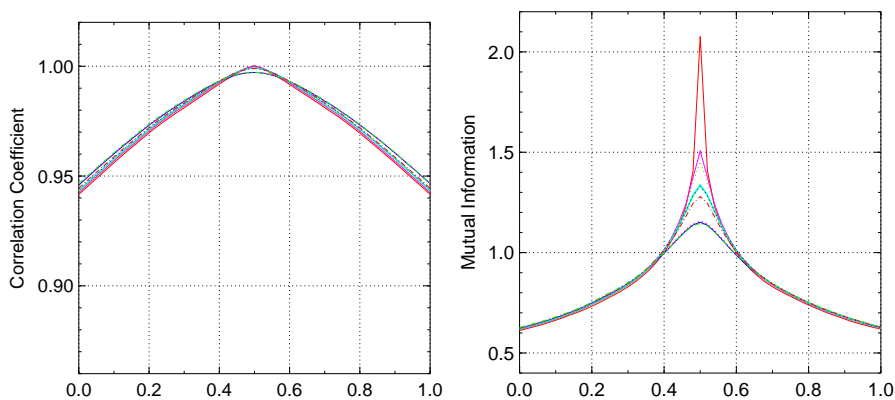


Figure 5.4: Similarity measure for rotations around the x axis

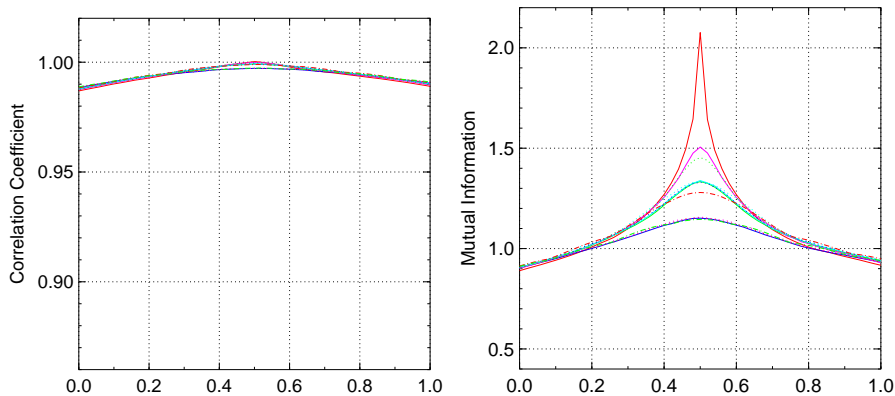


Figure 5.5: Similarity measure for rotations around the y axis

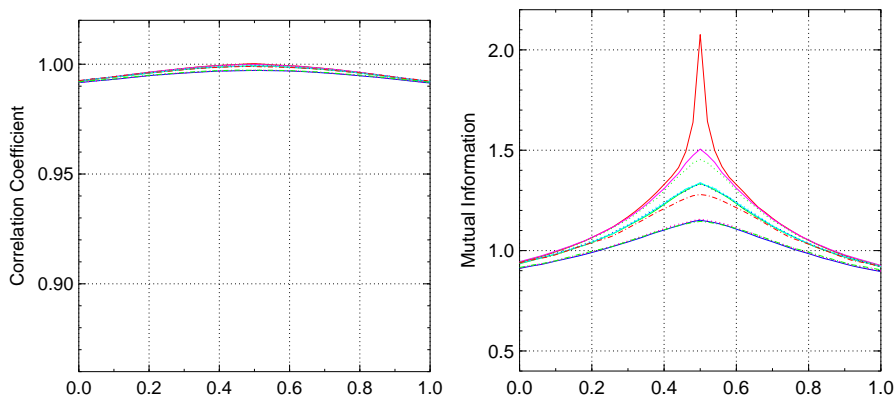


Figure 5.6: Similarity measure for rotations around the z axis

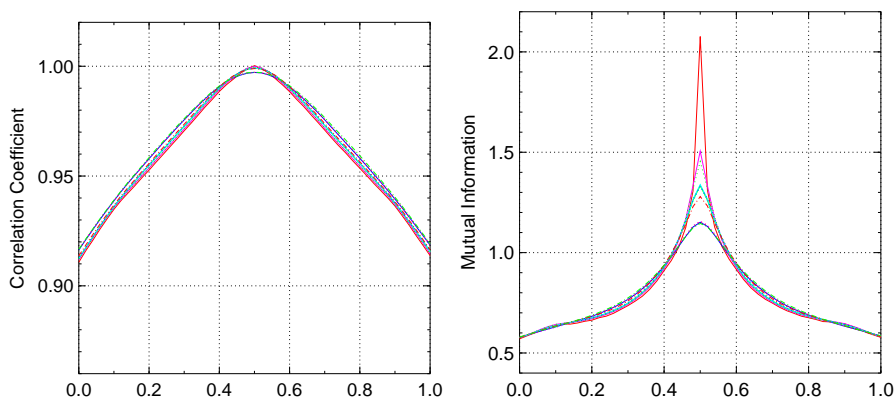


Figure 5.7: Similarity measure for translations along the y and z axes

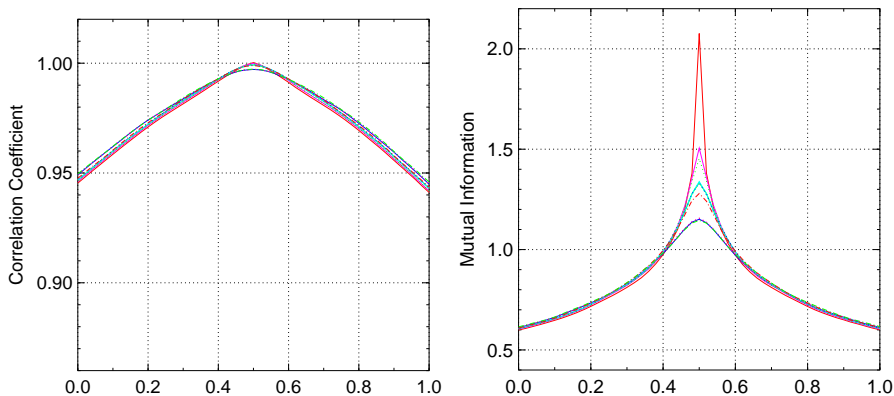


Figure 5.8: Similarity measure for rotations around the x and y axes

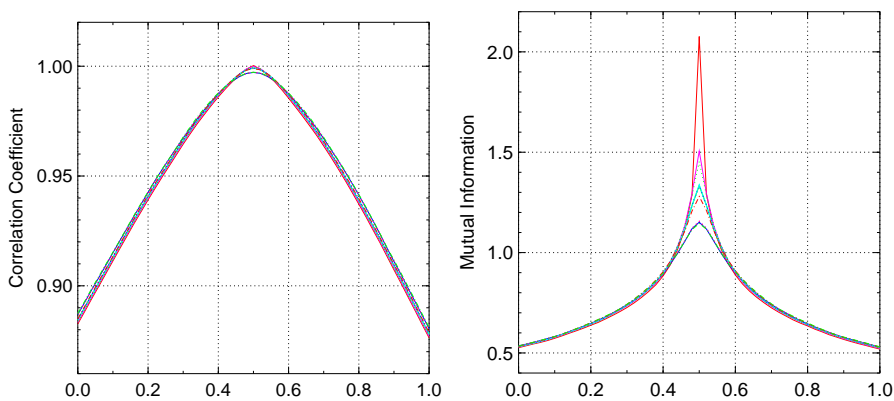


Figure 5.9: Similarity measure for translations along the y and z axes and rotations around the x and y axes

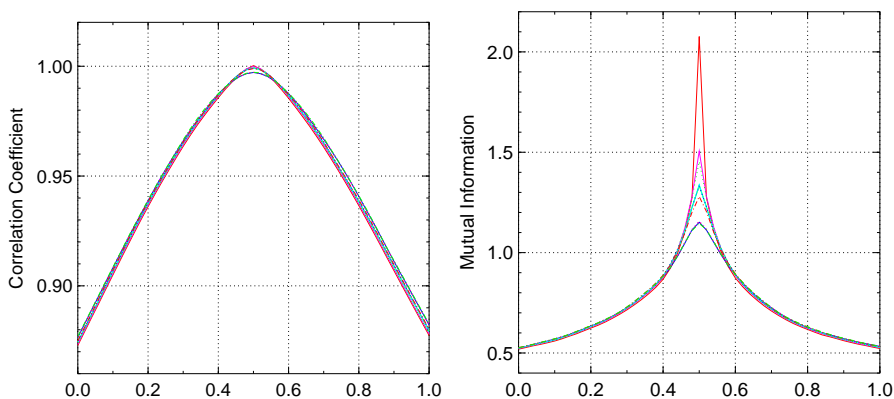


Figure 5.10: Similarity measure for translations and rotations on all the axes

5.1.1 Similarity measure performance

In most of the tests both the Correlation Coefficient as well as the Mutual Information similarity measures performed quite well in terms of the criteria we set out to meet initially. The curves are smooth and it does not contain local minima or maxima (extrema). Also the locations of the extrema coincide with the extremum of the ray casted DRRs.

A few exceptions are worth noting. The first is figure 5.1 where it is evident that the curve for the Correlation Coefficient similarity measure might be problematic, since it is very close to a straight line. Figure 5.11 shows a scaled version of the Correlation Coefficient graph in figure 5.1. It can be seen that the measure contains local extrema and that the the extrema of the different curves are not all aligned. A closer look reveals that the values lie in the range $[0.995 \dots 1.00025]$. This is an extremely small range and makes this similarity measure extremely sensitive to numerical errors introduced by the algorithm. Figures 5.5 and 5.6 suffers the same drawback. Other orientations of the CT cube also produced similarity curves with extremely small ranges when the Correlation Coefficient similarity measure is used. Figure 5.12 shows the results for two of these orientations.

Note that this behaviour is not restricted to translations on the x axis. However, translations on the x axis induces a zoom or shrink affect on the resulting image and it is evident that the Correlation Coefficient is not suitable for these cases. The Mutual Information similarity measure performed extremely well under all of the tests.

5.1.2 Light slab sampling

From the graphs it can be seen that light slabs with a higher sampling resolution of the virtual image plane mostly corresponds better with the ray casted DRR similarity curve. The light slab that performs best is the one with a virtual image plane sampling resolution of 512×512 and a focal plane sampling resolution of 32×32 , as indicated by the *solid magenta* line. The second best performing light slab has a virtual image plane sampling resolution of 512×512 and a focal plane resolution of 16×16 , as indicated by the *dotted green* line. However, the correspondence of the light slab with a virtual image plane sampling resolution of 512×512 and a focal plane resolution of 8×8 is not always third on the list of best performing light slabs, as indicated by the *dotted cyan* line. This is definitely attributed to the sampling of the focal plane being too low.

Since the virtual image plane size is not much larger than the computed DRR size it does not make sense to increase the virtual image plane sampling resolution to much higher than the DRR sampling resolution. Also, the focal plane size is small relative to the virtual image plane size and therefore a lower sampling resolution can still give favourable results.

5.2 Light slab generation time

The computation time of light slabs with different sampling dimensions on the image and focal planes were measured and tabulated. Although the light slab computation time is performed pre-operatively and therefore does not add to the patient treatment time, it is informative from a practical point of view to show that these computation times are not excessive. The parallelised light slab generation algorithm was used to generate the light slabs. The average computation time of the light slabs are given in table 5.2.

The differences between the minimum and maximum light slab computation times can be attributed to the fact that the ray cast algorithm, which is used to generate the light slab, has a computation time that is dependent on the complexity of the scene. For certain configurations, most notably a DRR with a view diagonally through the CT cube, the number of voxel traversals is significantly higher than other configurations.

The maximum time measured in this experiment is approximately 7 minutes for the light slab with a virtual image plane resolution of 256×256 and a focal plane resolution of 64×64 . Being a pre-operative computation, it is completely acceptable. Even if the image plane resolution is increased to 512×512 and the number of samples increases by a factor 4, the approximate computation time of 28 minutes is still acceptable.

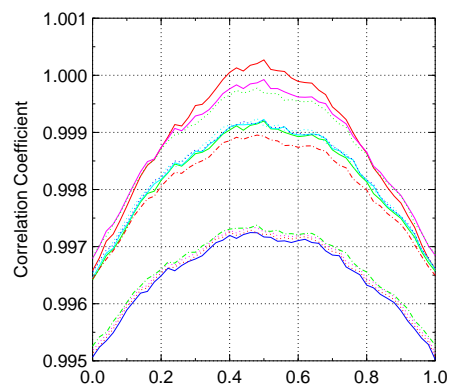


Figure 5.11: A scaled version of the Correlation Coefficient similarity measure for translations along the x axis.

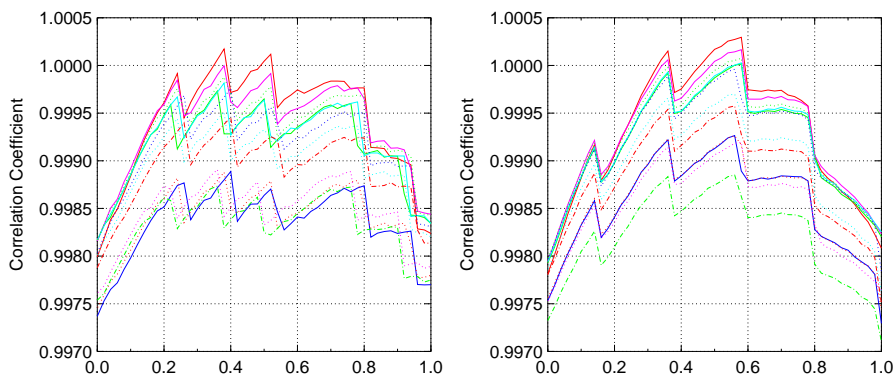


Figure 5.12: Similarity measure for translations along the x axis for two other CT orientations.

Focal Plane	Image Plane	Min (s)	Max (s)	Avg (s)
8 × 8	128 × 128	0.668	2.584	1.390
8 × 8	256 × 256	2.570	6.865	4.541
8 × 8	512 × 512	10.140	20.992	16.293
16 × 16	128 × 128	2.695	9.933	5.593
16 × 16	256 × 256	10.272	27.462	18.173
16 × 16	512 × 512	40.336	81.945	64.591
32 × 32	128 × 128	10.352	38.106	21.494
32 × 32	256 × 256	41.063	108.216	72.302
32 × 32	512 × 512	160.915	338.579	260.800
64 × 64	128 × 128	43.112	156.291	87.766
64 × 64	256 × 256	163.989	418.143	285.329

Table 5.2: The computation times of different light slabs.

5.3 DRR generation time

To measure the DRR generation time, a dense set of projections randomly distributed around a chosen reference position is computed. This is typically what the image registration optimiser would do. Using the ray casting algorithm, we generate the DRRs corresponding to the projection points and we repeat this using the light slab algorithm. The time taken to generate each DRR is measured for varying DRR resolutions and with parallelisation enabled and disabled, respectively. The measurement is started before the first pixel is computed and stopped after the last pixel is completed. Note that the light slab parameterisation theoretically does not affect the generation time of the light field DRRs.

The above mentioned procedure is repeated for a number of reference positions and the results shown in tables 5.3 to 5.5.

The ray cast algorithm is dependent on the complexity of the scene. That is, for different orientations one expects different computation times. On the other hand the light field algorithm computation is *not* dependent on the orientation. It simply performs an interpolation calculation to determine the value of each pixel. One therefore theoretically expects constant computation times for the various DRR resolutions. The results in tables 5.3 to 5.5 reflect these expectations. The minor variances in the computation times when using the light field algorithm can be attributed to other processes on the computer that required processing time.

Tables 5.4 and 5.5 show an average ~ 50 times speed increase when using the parallelised light field algorithm over the serial ray cast algorithm. The average increase in table 5.3 is slightly lower at ~ 47 times, but this can be attributed to the measurement granularity in the parallelised light field algorithm time measurements.

Sample	Ray Cast	Light Field	Parallel Ray Cast	Parallel Light Field
1	0.184	0.028	0.037	0.007
2	0.183	0.028	0.031	0.004
3	0.188	0.028	0.032	0.004
4	0.190	0.028	0.032	0.004
5	0.188	0.028	0.032	0.004
6	0.189	0.028	0.032	0.004
7	0.189	0.028	0.032	0.004
8	0.187	0.028	0.032	0.004
9	0.188	0.028	0.031	0.004
10	0.188	0.028	0.032	0.004
<i>avg</i>	0.187	0.028	0.032	0.004
σ	0.002	0.000	0.002	0.001

Table 5.3: Generation time (in seconds) of a 128×128 DRR using the various algorithms

Sample	Ray Cast	Light Field	Parallel Ray Cast	Parallel Light Field
1	0.700	0.111	0.114	0.014
2	0.696	0.111	0.114	0.014
3	0.711	0.111	0.115	0.014
4	0.716	0.110	0.116	0.014
5	0.712	0.111	0.116	0.014
6	0.712	0.111	0.117	0.014
7	0.714	0.111	0.117	0.014
8	0.709	0.110	0.116	0.014
9	0.711	0.111	0.114	0.014
10	0.709	0.111	0.116	0.014
<i>avg</i>	0.709	0.111	0.115	0.014
σ	0.006	0.000	0.001	0.000

Table 5.4: Generation time (in seconds) of a 256×256 DRR using the various algorithms

Sample	Ray Cast	Light Field	Parallel Ray Cast	Parallel Light Field
1	2.746	0.441	0.446	0.057
2	2.734	0.438	0.443	0.056
3	2.788	0.440	0.449	0.057
4	2.809	0.438	0.453	0.056
5	2.791	0.439	0.451	0.056
6	2.791	0.437	0.454	0.056
7	2.796	0.441	0.454	0.057
8	2.782	0.437	0.451	0.056
9	2.791	0.439	0.445	0.057
10	2.780	0.438	0.452	0.056
<i>avg</i>	2.781	0.439	0.450	0.056
<i>σ</i>	0.023	0.001	0.004	0.001

Table 5.5: Generation time (in seconds) of a 512×512 DRR using the various algorithms

5.4 Optimisation algorithm performance tests

In this experiment we evaluate the effect of using light field DRRs in an image registration algorithm compared to ray casted DRRs. Each algorithm is tested in conjunction with the two similarity measures and as in the similarity tests, a comparison will be made between solutions found using ray casted DRRs and those found using the light field algorithm.

Since we construct the experiment, we know the exact solution of the problem and can therefore compute the difference between the expected and calculated answers. Equation (4.1.4) is used to quantify the total error.

Van der Bijl (2006) suggested that the tolerance for the line search method used by Powell's algorithm (Brent's minimiser) be set to 0.1. No tolerance was set on the value of the cost function. The algorithm terminates when the absolute difference in all the individual dimensions of the solution are less than 0.1mm or 0.1° from the previous solution.

Figure 5.13 shows the images that the optimiser had to search for in the test cases. Tables 5.6 to 5.15 was generated using a light slab with a focal plane resolution of 64×64 and an image plane resolution of 256×256 . It contains the following information:

S The similarity measure used. The two options are:

M Mutual Information

C Correlation Coefficient

A The DRR generation algorithm used. The two options are:

R Ray Cast

L Light Field

ξ The total error as defined in equation (4.1.4), which is mainly used to quickly find outliers.

$\delta R_x, \delta R_y, \delta R_z$ The error in the rotation components measured as a percentage of the search space, which is computed using equation (4.1.3) with $i = 1, 2, 3$.

$\delta t_x, \delta t_y, \delta t_z$ The error in the translation components measured as a percentage of the search space, which is computed using equation (4.1.3) with $i = 4, 5, 6$.

N_f The number of cost function evaluations required by the image registration process.

N_D The number of DRR generations required by the image registration process.

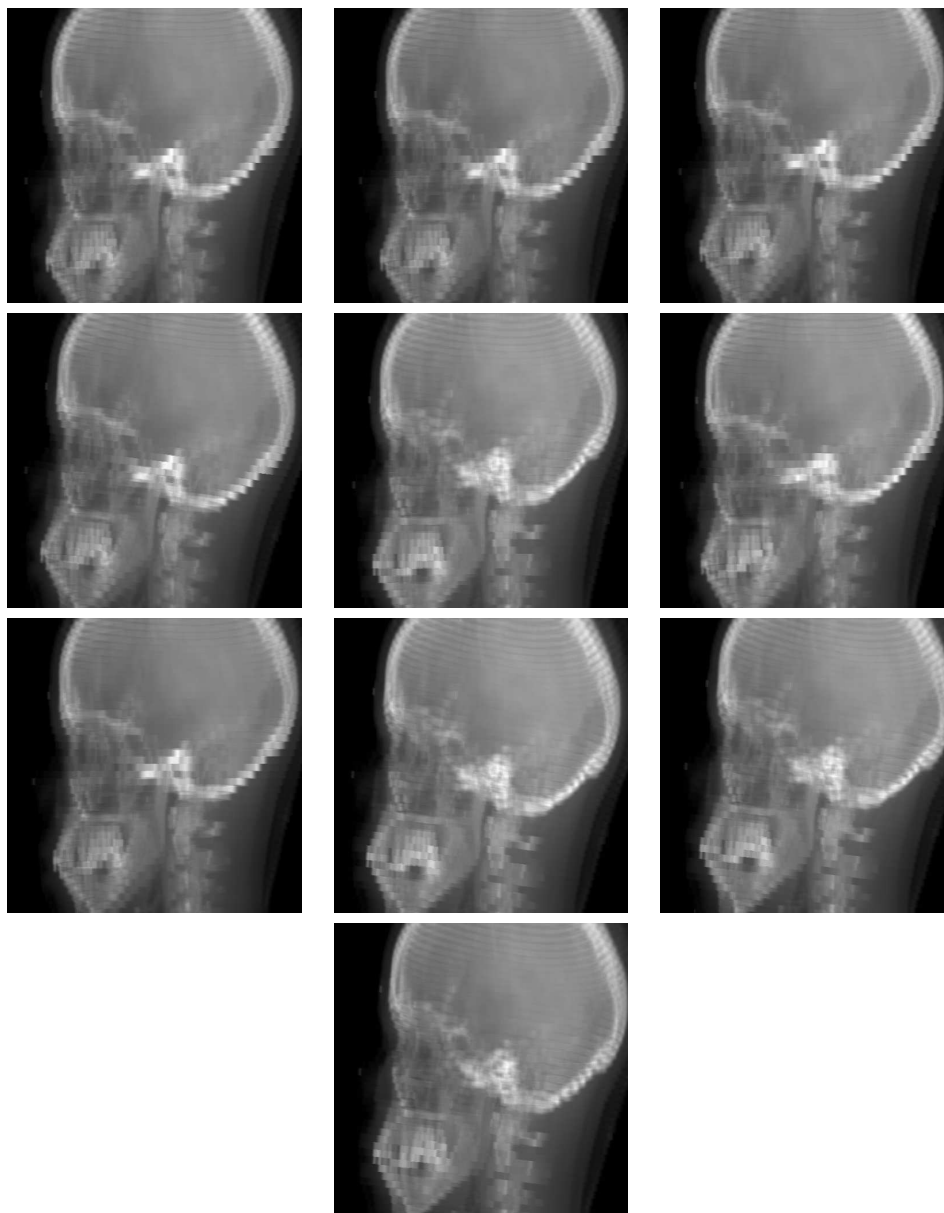


Figure 5.13: The perturbed images used in the optimiser tests. The images in the first row contains perturbations in δ_x , δ_y and δ_z , respectively. The second row contains rotation perturbations θ_x , θ_y and θ_z , respectively. The first image in the third row contains a combined perturbation in δ_y and δ_z , the second a combined perturbation in θ_y and θ_z and the third is a combination of the two aforementioned images. The image in the fourth row contains rotation and translation perturbations on all the axes.

Time The time (in seconds) required by the image registration process.

It is important to note that since the sizes of the search spaces are 10mm ($-5\text{mm} \dots 5\text{mm}$) and 10° ($-5^\circ \dots 5^\circ$) for translations and rotations, respectively, a one percent error in one of the dimensions translates to a real error of 0.1mm and 0.1° .

The SPG system aims to achieve sub-millimetre accuracy. This is the aim of the image registration system as well, which means that errors larger than 10% are considered unacceptably high.

The tests were performed using various light slab dimensions. Also, solutions were varied to lie within the sampled space and at the discontinuities in the cost function, which are boundary cases. The results corresponded and an arbitrary set is provided here. Appendix C contains three more result sets. The set provided here used a light slab with a focal plane resolution of 32×32 , and image plane resolution of 512×512 and the DRRs generated had a resolution of 512×512 . The optimum was located at +3mm and $+3^\circ$ from the planned position.

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.18	0.00	0.43	0.00	0.11	0.00	0.00	167	147	15.16
M	R	0.29	0.00	0.67	0.00	0.23	0.00	0.00	170	150	52.15
C	L	1.97	0.00	0.00	0.05	4.82	0.00	0.00	114	101	7.53
C	R	1.68	0.00	0.00	1.50	3.82	0.00	0.00	113	100	31.99

Table 5.6: Optimiser performance for a perturbation in δ_x

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.41	0.20	0.09	0.05	0.83	0.45	0.27	192	150	15.61
M	R	0.35	0.07	0.37	0.60	0.21	0.27	0.34	266	207	71.33
C	L	1.88	0.02	0.16	0.23	4.56	0.59	0.02	212	161	12.11
C	R	0.35	0.20	0.41	0.23	0.25	0.53	0.35	284	209	66.73

Table 5.7: Optimiser performance for a perturbation in δ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.12	0.00	0.02	0.24	0.15	0.02	0.08	251	199	20.57
M	R	0.34	0.08	0.34	0.57	0.08	0.09	0.48	257	205	71.23
C	L	0.87	0.58	0.67	0.78	1.75	0.15	0.07	270	204	15.28
C	R	2.93	0.16	0.34	0.40	7.14	0.05	0.12	276	209	66.98

Table 5.8: Optimiser performance for a perturbation in δ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.29	0.03	0.09	0.24	0.66	0.05	0.10	137	125	12.86
M	R	0.38	0.75	0.48	0.00	0.13	0.24	0.07	61	56	19.42
C	L	2.18	0.44	0.50	0.01	5.27	0.24	0.55	123	110	8.19
C	R	2.14	0.27	0.06	0.00	5.18	0.64	0.32	60	54	17.18

Table 5.9: Optimiser performance for a perturbation in θ_x

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.12	0.00	0.29	0.00	0.00	0.00	0.00	112	99	10.14
M	R	0.20	0.00	0.49	0.00	0.00	0.00	0.00	57	51	17.73
C	L	1.14	0.00	0.68	0.00	2.72	0.00	0.00	112	99	7.39
C	R	1.89	1.22	0.11	0.00	4.46	0.00	0.00	56	50	16.00

Table 5.10: Optimiser performance for a perturbation in θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.13	0.00	0.00	0.31	0.00	0.00	0.00	112	99	10.16
M	R	0.16	0.00	0.00	0.38	0.00	0.00	0.00	57	51	17.98
C	L	0.19	0.00	0.00	0.47	0.00	0.00	0.00	57	51	3.79
C	R	0.18	0.00	0.00	0.44	0.00	0.00	0.00	55	49	15.96

Table 5.11: Optimiser performance for a perturbation in θ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.45	0.46	0.12	0.17	0.91	0.30	0.27	262	203	21.10
M	R	0.34	0.11	0.31	0.25	0.08	0.59	0.39	330	261	89.89
C	L	1.44	0.33	0.31	0.06	3.46	0.45	0.32	282	203	15.21
C	R	0.40	0.47	0.55	0.60	0.30	0.05	0.05	317	214	68.35

Table 5.12: Optimiser performance for perturbations in δ_y and δ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.43	0.84	0.02	0.29	0.51	0.00	0.21	189	169	17.41
M	R	0.27	0.61	0.16	0.01	0.00	0.18	0.12	60	56	19.41
C	L	0.56	0.35	0.00	0.46	1.07	0.30	0.56	115	104	7.74
C	R	0.52	0.16	0.01	0.36	1.00	0.46	0.54	59	56	17.81

Table 5.13: Optimiser performance for perturbations in θ_x and θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.37	0.45	0.02	0.34	0.27	0.43	0.48	259	202	21.07
M	R	0.27	0.10	0.28	0.34	0.11	0.48	0.05	347	273	94.29
C	L	3.36	0.53	0.11	0.04	8.19	0.43	0.40	224	165	12.38
C	R	0.60	0.49	0.87	0.91	0.35	0.43	0.14	209	160	51.74

Table 5.14: Optimiser performance for perturbations in δ_y , δ_z , θ_x and θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.35	0.04	0.52	0.13	0.65	0.08	0.18	395	282	29.19
M	R	0.36	0.12	0.51	0.23	0.52	0.44	0.05	295	244	86.63
C	L	1.83	0.05	0.11	0.09	4.44	0.56	0.18	249	194	14.51
C	R	1.17	0.31	0.27	0.03	2.80	0.26	0.27	251	195	63.43

Table 5.15: Optimiser performance for perturbations in δ_x , δ_y , δ_z , θ_x , θ_y , θ_z

5.4.1 Performance in terms of accuracy

The performance of the Mutual Information similarity measure in terms of accuracy is very good. The maximum error found on an individual axis was 3.94% (table C.10), which translates to 0.394mm. In most cases, however, the errors on the individual axes were less than 1%.

The Correlation Coefficient similarity measure performs really badly when determining the x translation parameter. This is not unexpected. In section §5.1.1 we already saw that the Correlation Coefficient similarity measure performed poorly and this reflects directly in the optimiser performance. A maximum error of 14.54% (table C.7) was measured for the x translation parameter. Although this error is unacceptably high, it is interesting to note that in practice the x translation is the least important positioning parameter. It has the effect that it moves the patient closer or farther away from the beam. Since air does not have a great impact on the beam, having patient closer or farther from the beam source does not greatly affect the clinical aspects of the treatment.

If we disregard the x translation parameter we have a maximum error of 2.33% (table C.22) which is almost half of the worst case performance of the Mutual Information similarity measure. In most cases the errors on the individual axes were less than 1%, similar to the Mutual Information similarity measure.

For interest's sake we note that the maximum error found when using the Mutual Information similarity measure and disregarding the x translation parameter, was 1.82% (table C.17).

All tests using the light field algorithm performed well compared to the tests using the ray cast algorithm, with accuracy that was mostly in the same order of magnitude and sometimes even better.

5.4.2 Performance in terms of computation time

In all the experiments the optimiser using the Correlation Coefficient similarity measure combined with DRRs generated from light slab performed the fastest. The maximum time recorded for this setup was 20.31 seconds (table C.2). The second fastest setup in all experiments are the Mutual Information similarity measure combined with DRRs generated from light slabs. The maximum time recorded was 39.36 seconds (table C.20). This is almost double the worst case time when using the Correlation Coefficient similarity measure, but still a big improvement over the time taken for manual verification. It is important to note that in all experiments the optimisation processes using the light field algorithm always outperformed those using ray casting.

5.4.3 Boundary cases

When the optimum is located on the edge of the sampled space, that is, close to the discontinuities in the cost function, the optimisation algorithms in some cases take excessively long to complete. One such an exception can be seen in table C.19. This outlier, however, is not due to the use of the light field algorithm, as the ray cast algorithm also produced two outliers shown in tables C.20 and C.30. These anomalies never occurred for the Correlation Coefficient measure and therefore are restricted to the Mutual Information similarity measure.

In all three of the abovementioned anomalies the number of cost function evaluations N_f is excessively high. Furthermore, the number of cost function evaluations is very high in relation to the number of DRRs generated, N_D . This is an indication that the optimising algorithm did not steadily converge to the optimum, but repeatedly searched outside the search space of the light slab. This may be attributed to Brent's algorithm trying to do a quadratic fit using points situated around the discontinuity when the Mutual Information similarity measure is used. It is suggested that the cost function c be redefined in an attempt to alleviate the impact of the discontinuities found at ϵ and $-\epsilon$ on the optimiser. Note that the anomalies were not observed when the optima was located at or further than 0.5mm or 0.5° from a discontinuity, as shown in section §C.1.

In practice an upper limit can be set on the number of cost function evaluations that are allowed. If more than this number of steps is required by the optimiser the algorithm will terminate without having completed the registration process. The registration process can then be restarted using the Correlation Coefficient similarity measure. Also, the chosen error limits are generous, so the actual positioning error should be significantly smaller than 5mm and 5° , which implies that the optima should never be near the discontinuities in practice.

5.5 Summary

The tests conducted in this chapter showed that DRRs generated using the light field algorithm are useable in an image registration process. In particular it was determined that an image registration system using these DRRs is robust, accurate and fast, which are the aims defined at the start of this thesis.

The equivalence of the similarity curves generated using DRRs calculated using the light field algorithm and DRRs calculated using the ray cast algorithm was shown to be dependent on the light slab sampling densities. In most cases the light field curves corresponded to the ray cast similarity curves, the only exceptions being numerical errors introduced when the range of the curve was extremely small.

Light slab generation is done pre-operatively and does not affect the patient treatment time, but it was shown that the time taken to generate the light slabs are realistic to be performed in practice.

When examining the results for the time performance of the optimisation process, three anomalies were observed when the Mutual Information similarity measure was used. These anomalies were attributed to a suboptimal unconstrained cost function definition.

The tests reported all used the CT cube of a real patient. CT cubes of other patients behaved similarly. No compression of the light slab was implemented. The optimiser was shown to work for a light slab with a focal plane resolution of 32×32 and a virtual image plane resolution of 512×512 (sections §5.4 and §C.2) as well as a light slab with a focal plane resolution of 64×64 and a virtual image plane resolution of 256×256 (sections §C.1 and §C.3). The memory requirement for each light slab was 2^{30} bytes, which translates to 1GB (one gigabyte).

Chapter 6

Conclusions and recommendations

At the start of this thesis we set out to find DRR generation algorithm that is fast and that, when used in conjunction with an image registration process, produces a fast, accurate and robust method for verifying the patient position. The goal was to be able to accurately determine the error in the patient position in under three minutes.

Various DRR generation algorithms were investigated. Image-order algorithms like the ray cast algorithm and the light field algorithm were shown to be more feasible for parallelisation than object-order algorithms, like splatting or voxel projection, as well as hybrid algorithms like cylindrical harmonics and the shear-warp algorithm.

A parallelised implementation of the light field algorithm was shown to satisfy all the requirements. Accurate registration is performed in under a minute and the algorithm will automatically perform even better if more CPUs are added to the machine on which it is executed. This greatly improves the 7.5 minutes reported by van der Bijl (2006). The parallelised light field algorithm performs roughly 50 times faster than the serial ray cast algorithm as proposed by Jacobs *et al.* (1998). This improvement did not directly translate to the optimiser time, as the implementation in this thesis required substantially more cost function evaluations than was the case in van der Bijl (2006).

An arbitrary definition of an unconstrained cost function was used in the image registration implementation. This cost function was shown to work with both the Mutual Information and the Correlation Coefficient similarity measures. Although the Mutual Information similarity measure took longer to complete than the Correlation Coefficient similarity measure in some cases, it produced lower errors overall as it does not suffer from small ranges in the similarity measure like the Correlation Coefficient does. It did, however, suffer from a few cases where the optimiser took excessively long to compute the optimum. The definition of the unconstrained cost function and possibly the

parameters used by Brent's line minimisation would be worthwhile areas for further investigations.

In terms of accuracy, the maximum error when using the Mutual Information similarity measure was 3.94%. This translated to 0.394mm. Most of the errors were smaller than 1%, which implies that the errors were smaller than 0.1mm and 0.1° . The Correlation Coefficient performed badly when determining the x translation component in some tests. The maximum error was 14.54%. This was expected, since the range of the similarity curve was very small. When the x translation components are disregarded the maximum error was 2.33% (0.233°), whilst most of the errors on the individual axes were smaller than 1%.

The similarity measures work accurately even if a large portion of the image does not contain tissue data. For our test purposes it was therefore acceptable to disregard segmentation. In practice, however, segmentation will be used to reduce the chance of artifacts on the PR such as the patient positioning gear being introduced in the comparison with the DRRs.

Appendix A

System specifications

A.1 Hardware

- CPUs: Two quad core Intel(R) Xeon(R) 2.00GHz CPUs
- RAM: 2GB

A.2 Software

- Kernel: Linux 2.6.18-53.1.14.el5 #1 SMP Wed Mar 5 10:08:25 EST 2008
x86_64
- Compiler: gcc version 4.1.2 20070626 (Red Hat 4.1.2-14)
- VXL version 1.10.0: A bug in the Powell optimiser was fixed and submitted as a correction and will be available in VXL 1.11.
- The ray cast and light field algorithms were implemented in C++.

Appendix B

Mathematical aspects

B.1 Rotation matrices

Counterclockwise coordinate system rotations when looking towards the origin from a positive axis, gives the matrices:

$$\begin{aligned} R_x(\alpha) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \\ 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix} \\ R_y(\beta) &= \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \\ R_z(\gamma) &= \begin{bmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \end{aligned}$$

Rotation matrices are by definition orthogonal and thus have the properties that $R^{-1} = R^T$ and $\det R = 1$.

B.2 Transformations

Transformations are made up of two parts: a rotation matrix and a translation vector. We use the notation $G_{BA} = (R_{BA}, \mathbf{t}_{BA})$ to denote the transformation from coordinate system \mathcal{A} to coordinate system \mathcal{B} . More specifically, a vector \mathbf{p}_A in coordinate system \mathcal{A} is transformed to coordinate system \mathcal{B} by

$$\mathbf{p}_B = R_{BA}\mathbf{p}_A + \mathbf{t}_{BA} .$$

When one transformation is applied to another transformation, the resultant transformation is given by

$$\begin{aligned} G_{CB} \diamond G_{BA} &= (R_{CB}, \mathbf{t}_{CB}) \diamond (R_{BA}, \mathbf{t}_{BA}) \\ &= (R_{CB}R_{BA}, R_{CB}\mathbf{t}_{BA} + \mathbf{t}_{CB}) \\ &= G_{CA} \quad . \end{aligned}$$

The *inverse* of a transformation G_{BA} is the transformation G_{AB} , given by

$$G_{AB} = (R_{AB}, \mathbf{t}_{AB}) = (R_{BA}^{-1}, -R_{BA}^{-1}\mathbf{t}_{BA}) \quad .$$

B.3 Determining the intersection of a line with a plane

Consider a line crossing through two points, \mathbf{p}_1 and \mathbf{p}_2 . The parametrised equation for any point \mathbf{p} on that line is given by

$$\mathbf{p} = \mathbf{p}_1 + \alpha(\mathbf{p}_2 - \mathbf{p}_1) \quad . \quad (\text{B.3.1})$$

To find the intersection of this line with an arbitrary plane defined by $x = k$, one simply compute α from equation (B.3.1):

$$\begin{aligned} k &= x_1 + \alpha(x_2 - x_1) \\ \alpha &= \frac{k - x_1}{x_2 - x_1} \quad . \end{aligned}$$

The solutions of the y and z components then follow directly from equation (B.3.1).

B.4 Mutual Information properties

The Mutual Information similarity measure has the following properties:

- It is symmetric. That is

$$MI(A, B) = MI(B, A)$$

for images A and B .

- The information an image contains about itself is equal to the entropy of the image, so

$$MI(A, A) = H(A) \quad .$$

- The information images contain about each other can never be greater than the information in the images themselves.

$$MI(A, B) \leq H(A)$$

$$MI(A, B) \leq H(B)$$

- The uncertainty about A cannot be increased by learning about B .

$$MI(A, B) \geq 0$$

- When A and B are not related, no knowledge is gained about one image when the other is given, so

$$MI(A, B) = 0$$

if and only if A and B are independent.

B.5 Steradians

A *solid angle* is the 3D analogue of a 2D angle. Where an angle is bounded by two lines, a solid angle is bounded by the generators of a cone. A solid angle is measured in steradians, which is defined to be the area of the intersection of a solid angle with a unit sphere (Clapham, 1996). The word *steradian* is derived from the Greek word *stereos* for "solid" and the Latin word *radius* for "ray" or "beam" (Wikipedia, 2008).

An angle (measured in radians) is related to the arc length it cuts out in the following way:

$$\theta = \frac{s}{r} \quad ,$$

where s is the arc length and r the radius of the circle. A solid angle (measured in steradians) is related to the area it cuts out, as follows:

$$\Omega = \frac{S}{r^2} \quad ,$$

where S is the surface area, and r is the radius of the sphere.

The steradian is dimensionless, since $1sr = m^2 \times m^{-2} = 1$. In practice the symbol "sr" is used where appropriate, in order to distinguish between dimensionless quantities of different natures.

B.6 Taylor approximations

The Taylor series of function $f(x)$ that is infinitely differentiable in a neighbourhood of a , is the power series

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots \quad ,$$

which can also be written as

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n \quad ,$$

where $n!$ is the factorial of n and $f^{(n)}(a)$ denotes the n th derivative of f evaluated at the point a . Note that the 0th derivative of f is defined to be f itself and $(x-a)^0$ and $0!$ are both defined to be 1.

Appendix C

Additional optimiser performance results

C.1 Focal plane 64×64 , image plane 256×256 , DRB 512×512 and optimum located 0.5mm and 0.5° from $-\epsilon$

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.25	0.04	0.08	0.05	0.57	0.12	0.14	188	140	14.15
M	R	0.26	0.18	0.01	0.10	0.39	0.21	0.39	199	151	52.10
C	L	2.90	0.57	0.27	0.58	7.03	0.36	0.33	122	94	6.93
C	R	0.85	0.07	0.27	0.09	2.04	0.10	0.21	127	99	31.54

Table C.1: Optimiser performance for a perturbation in δ_x

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.39	0.22	0.11	0.44	0.65	0.45	0.12	261	191	19.45
M	R	0.52	0.09	0.09	0.02	1.18	0.22	0.44	411	307	108.10
C	L	2.49	0.32	0.22	1.25	5.96	0.16	0.19	395	274	20.31
C	R	3.95	0.15	0.02	0.27	9.66	0.35	0.05	366	260	85.44

Table C.2: Optimiser performance for a perturbation in δ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.39	0.12	0.15	0.08	0.89	0.11	0.29	261	193	19.68
M	R	0.37	0.08	0.01	0.63	0.56	0.28	0.21	205	153	53.08
C	L	1.60	0.42	0.69	1.41	3.57	0.07	0.17	198	144	10.74
C	R	0.28	0.33	0.37	0.18	0.41	0.14	0.01	262	188	60.03

Table C.3: Optimiser performance for a perturbation in δ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.32	0.26	0.44	0.52	0.11	0.06	0.23	138	101	10.25
M	R	0.33	0.42	0.03	0.33	0.35	0.46	0.16	75	57	19.66
C	L	1.54	0.18	0.13	1.27	3.49	0.23	0.50	138	101	7.48
C	R	2.01	0.18	0.10	0.50	4.89	0.14	0.02	73	55	17.55

Table C.4: Optimiser performance for a perturbation in θ_x

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.27	0.03	0.52	0.05	0.37	0.16	0.07	123	90	9.12
M	R	0.25	0.00	0.23	0.44	0.04	0.00	0.38	63	49	17.05
C	L	2.23	0.54	1.07	0.17	5.32	0.21	0.08	125	92	6.83
C	R	1.78	0.74	1.21	0.81	3.91	0.05	1.04	126	94	30.19

Table C.5: Optimiser performance for a perturbation in θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.59	0.00	0.02	0.22	1.42	0.13	0.15	187	137	13.79
M	R	0.30	0.38	0.37	0.25	0.36	0.28	0.00	134	103	37.16
C	L	0.66	0.53	0.44	0.42	1.23	0.15	0.65	124	92	6.80
C	R	0.28	0.00	0.00	0.23	0.60	0.09	0.19	61	49	16.20

Table C.6: Optimiser performance for a perturbation in θ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.52	0.19	0.91	0.30	0.55	0.45	0.40	273	201	20.63
M	R	0.31	0.08	0.30	0.14	0.29	0.49	0.36	285	216	76.02
C	L	5.95	0.33	0.37	0.47	14.54	0.26	0.37	301	209	15.52
C	R	0.34	0.10	0.42	0.35	0.19	0.24	0.52	306	214	70.43

Table C.7: Optimiser performance for perturbations in δ_y and δ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.33	0.32	0.60	0.29	0.25	0.04	0.17	209	150	15.18
M	R	0.67	0.47	0.49	0.15	1.35	0.42	0.42	222	163	56.41
C	L	1.06	0.23	0.81	1.03	2.22	0.22	0.19	141	101	7.49
C	R	1.79	0.39	0.35	0.11	4.33	0.18	0.08	140	101	32.21

Table C.8: Optimiser performance for perturbations in θ_x and θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.90	0.29	0.26	0.45	1.96	0.51	0.64	478	326	33.43
M	R	0.58	0.29	0.28	0.39	1.28	0.15	0.27	397	278	97.82
C	L	0.74	0.10	0.88	0.46	1.39	0.22	0.55	319	213	15.79
C	R	0.24	0.45	0.07	0.28	0.09	0.23	0.00	322	211	69.26

Table C.9: Optimiser performance for perturbations in δ_y , δ_z , θ_x and θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.26	0.24	0.26	0.20	0.42	0.19	0.21	423	267	27.30
M	R	1.67	0.53	0.35	0.10	3.94	0.83	0.05	415	274	98.73
C	L	1.50	1.07	0.21	0.19	3.22	0.29	1.35	226	157	11.62
C	R	1.03	0.47	0.02	0.15	2.41	0.06	0.49	236	161	53.15

Table C.10: Optimiser performance for perturbations in δ_x , δ_y , δ_z , θ_x , θ_y , θ_z

C.2 Focal plane 32×32 , image plane 512×512 , DRR 512×512 and optimum located at ϵ

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.15	0.04	0.11	0.30	0.00	0.16	0.03	194	152	15.57
M	R	0.45	0.55	0.57	0.07	0.33	0.57	0.40	284	216	73.64
C	L	0.62	0.00	0.00	0.50	1.14	0.09	0.86	118	99	7.33
C	R	2.31	0.00	0.00	1.40	5.48	0.22	0.05	117	101	31.44

Table C.11: Optimiser performance for a perturbation in δ_x

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.37	0.04	0.39	0.59	0.43	0.36	0.12	219	162	16.90
M	R	0.34	0.33	0.03	0.51	0.12	0.36	0.43	222	165	56.39
C	L	3.35	0.61	0.36	1.93	7.87	0.86	0.69	313	218	16.27
C	R	2.44	0.04	0.22	0.86	5.88	0.53	0.13	362	255	80.48

Table C.12: Optimiser performance for a perturbation in δ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.33	0.36	0.49	0.11	0.11	0.34	0.35	222	167	17.42
M	R	0.32	0.03	0.43	0.54	0.01	0.18	0.35	224	169	57.60
C	L	0.96	0.48	0.08	0.40	1.93	0.15	1.17	281	201	14.92
C	R	1.21	0.89	1.19	0.50	2.18	0.25	1.22	287	207	64.83

Table C.13: Optimiser performance for a perturbation in δ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.29	0.17	0.29	0.20	0.03	0.43	0.40	131	106	10.89
M	R	0.21	0.00	0.00	0.28	0.20	0.36	0.13	65	59	19.90
C	L	0.52	0.10	0.57	0.59	0.95	0.17	0.03	126	99	7.37
C	R	0.67	0.46	0.02	0.00	1.45	0.49	0.38	63	53	16.35

Table C.14: Optimiser performance for a perturbation in θ_x

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.25	0.00	0.00	0.18	0.11	0.31	0.49	120	97	9.95
M	R	0.42	0.19	0.00	0.55	0.71	0.05	0.49	209	162	55.25
C	L	2.26	0.00	0.46	0.20	5.39	0.23	1.13	121	97	7.18
C	R	2.50	0.00	0.00	0.59	6.00	0.07	1.04	121	98	30.66

Table C.15: Optimiser performance for a perturbation in θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.28	0.00	0.00	0.00	0.22	0.38	0.52	119	98	9.95
M	R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	57	50	17.56
C	L	2.24	0.00	0.00	0.46	5.42	0.22	0.76	119	97	7.13
C	R	2.36	0.00	0.00	0.00	5.79	0.00	0.00	58	51	16.45

Table C.16: Optimiser performance for a perturbation in θ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	1.61	0.29	0.41	0.88	2.87	1.72	1.82	491	333	34.78
M	R	0.77	0.24	0.63	0.61	0.12	1.51	0.68	322	227	77.29
C	L	0.77	0.51	0.15	0.11	0.02	0.81	1.62	312	210	15.63
C	R	2.17	0.09	0.51	0.39	5.20	0.80	0.49	313	209	66.18

Table C.17: Optimiser performance for perturbations in δ_y and δ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.25	0.46	0.00	0.31	0.18	0.14	0.17	145	110	11.32
M	R	0.33	0.46	0.00	0.00	0.62	0.27	0.03	65	57	19.31
C	L	2.09	0.26	0.00	0.16	5.02	0.01	0.89	144	109	8.07
C	R	0.25	0.17	0.00	0.10	0.26	0.08	0.51	142	107	33.24

Table C.18: Optimiser performance for perturbations in θ_x and θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	1.56	1.80	1.09	0.09	2.42	1.80	1.00	6218	3733	388.07
M	R	0.54	0.25	0.25	0.24	1.20	0.25	0.25	488	266	91.71
C	L	4.43	0.06	0.54	0.59	10.73	1.23	0.72	252	168	12.43
C	R	0.80	0.97	0.08	1.28	0.34	0.70	0.83	244	160	49.87

Table C.19: Optimiser performance for perturbations in δ_y , δ_z , θ_x and θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.99	1.04	1.51	0.38	0.04	0.18	1.52	662	373	39.36
M	R	0.92	1.03	0.47	1.16	0.77	0.74	1.13	979	571	206.09
C	L	1.58	1.11	0.78	1.15	3.23	0.38	1.11	367	227	16.87
C	R	0.99	0.48	0.58	0.96	1.65	0.67	1.11	350	213	68.59

Table C.20: Optimiser performance for perturbations in $\delta_x, \delta_y, \delta_z, \theta_x, \theta_y, \theta_z$

C.3 Focal plane 64×64 , image plane 256×256 , DRR 512×512 and optimum located at ϵ

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.05	0.00	0.00	0.00	0.00	0.11	0.05	115	96	9.89
M	R	0.58	0.00	1.00	1.00	0.00	0.00	0.00	57	50	17.02
C	L	0.31	0.00	0.00	0.00	0.76	0.00	0.00	59	52	3.81
C	R	2.31	0.00	0.00	1.40	5.48	0.22	0.05	117	101	31.46

Table C.21: Optimiser performance for a perturbation in δ_x

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.54	0.07	0.67	0.64	0.59	0.68	0.23	221	160	16.60
M	R	0.34	0.33	0.03	0.51	0.12	0.36	0.43	222	165	56.47
C	L	1.18	0.05	2.33	1.40	0.53	0.74	0.27	289	203	14.95
C	R	2.44	0.04	0.22	0.86	5.88	0.53	0.13	362	255	80.43

Table C.22: Optimiser performance for a perturbation in δ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.39	0.37	0.66	0.15	0.28	0.35	0.35	220	164	17.04
M	R	0.32	0.03	0.43	0.54	0.01	0.18	0.35	224	169	57.95
C	L	1.09	0.19	1.30	0.09	2.28	0.05	0.48	282	206	15.18
C	R	1.21	0.89	1.19	0.50	2.18	0.25	1.22	287	207	64.90

Table C.23: Optimiser performance for a perturbation in δ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.29	0.00	0.25	0.30	0.20	0.43	0.37	131	106	10.94
M	R	0.21	0.00	0.00	0.28	0.20	0.36	0.13	65	59	19.94
C	L	0.58	0.26	0.13	0.87	1.09	0.10	0.09	128	102	7.52
C	R	0.68	0.46	0.75	0.56	1.27	0.07	0.23	126	97	29.99

Table C.24: Optimiser performance for a perturbation in θ_x

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.27	0.18	0.00	0.02	0.52	0.17	0.34	201	154	15.82
M	R	0.42	0.19	0.00	0.55	0.71	0.05	0.49	209	162	55.36
C	L	3.17	0.21	0.84	0.25	7.70	0.20	0.09	120	98	7.24
C	R	2.50	0.00	0.00	0.59	6.00	0.07	1.04	121	98	30.79

Table C.25: Optimiser performance for a perturbation in θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.32	0.00	0.00	0.00	0.78	0.05	0.05	116	95	9.79
M	R	0.00	0.00	0.00	0.00	0.00	0.00	0.00	57	50	17.63
C	L	2.56	0.00	0.00	0.03	6.26	0.28	0.50	118	97	7.10
C	R	1.53	0.00	0.00	0.00	3.75	0.13	0.16	119	98	32.27

Table C.26: Optimiser performance for a perturbation in θ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	1.02	0.09	1.09	0.37	0.17	1.75	1.33	322	227	23.68
M	R	0.77	0.24	0.63	0.61	0.12	1.51	0.68	322	227	77.38
C	L	0.62	0.16	0.69	0.50	0.72	0.80	0.62	308	211	15.52
C	R	2.17	0.09	0.51	0.39	5.20	0.80	0.49	313	209	66.15

Table C.27: Optimiser performance for perturbations in δ_y and δ_z

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.36	0.36	0.34	0.61	0.37	0.06	0.08	230	165	17.11
M	R	0.40	0.84	0.38	0.14	0.26	0.14	0.05	146	111	37.71
C	L	2.94	0.10	0.00	1.51	7.04	0.17	0.06	141	107	7.90
C	R	0.25	0.17	0.00	0.10	0.26	0.08	0.51	142	107	33.28

Table C.28: Optimiser performance for perturbations in θ_x and θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.29	0.57	0.06	0.07	0.23	0.32	0.16	572	354	37.10
M	R	0.59	0.25	0.26	0.10	1.35	0.25	0.25	388	218	75.31
C	L	5.21	0.86	1.10	0.72	12.58	0.77	1.29	235	155	11.40
C	R	0.80	0.97	0.08	1.28	0.34	0.70	0.83	244	160	49.93

Table C.29: Optimiser performance for perturbations in δ_y , δ_z , θ_x and θ_y

S	A	ξ	δR_x	δR_y	δR_z	δt_x	δt_y	δt_z	N_f	N_D	Time
M	L	0.57	0.51	0.19	0.70	0.58	0.02	0.89	660	342	35.88
M	R	0.92	1.03	0.47	1.16	0.77	0.74	1.13	979	571	206.18
C	L	1.85	1.40	0.85	1.97	3.38	1.16	1.14	319	199	14.62
C	R	0.99	0.48	0.58	0.96	1.65	0.67	1.11	350	213	68.70

Table C.30: Optimiser performance for perturbations in $\delta_x, \delta_y, \delta_z, \theta_x, \theta_y, \theta_z$

Bibliography

- Amdahl, G. (1996 April). Validity of the single processor approach to achieving large-scale computing capabilities. In: *AFIPS Conference Proceedings*, vol. 30, pp. 483–485. AFIPS Press, Reston, Va., Atlantic City, N.I. (Cited on page 27.)
- Baumann, M. (2004 September). *Fast DRR Generation with Light Fields*. Diploma Thesis, Universitat Karlsruhe (TH). (Cited on page 45.)
- Camahort, E., Leros, A. and Fussell, D. (1998). Uniformly sampled light fields. In: Drettakis, G. and Max, N. (eds.), *Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98)*, pp. 117–130. Springer Wien, New York, NY.
Available at: <http://citeseer.ist.psu.edu/camahort98uniformly.html>
(Cited on page 44.)
- Carstens, C. and Muller, N. (2007). Fast calculation of digitally reconstructed radiographs using parallelism. In: Tapamo, J. and Nichols, F. (eds.), *Proceedings of the Eighteenth Annual Symposium of the Pattern Recognition Association of South Africa*, pp. 57–62. Pattern Recognition Association of South Africa, Durban, South Africa. ISBN 978-1-86840-656-2. (Cited on page 27.)
- Clapham, C. (1996). *The Concise Oxford Dictionary of Mathematics*. 2nd edn. Oxford University Press. (Cited on page 98.)
- de Kock, E.A. (2004 June). Concepts and definitions required for the development of a portal radiographic verification system at iThemba LABS. Tech. Rep.. (Cited on pages 2 and 4.)
- de Kock, E.A. (2008 June). Personal Communication. IThemba LABS. (Cited on pages 5 and 6.)
- Greig, D.M. (1980). *Optimisation*. Longman Inc., New York, NY, USA. ISBN 0-582-44186-2. (Cited on pages 56, 57, and 59.)
- Ihm, I., Park, S. and Lee, R.K. (1997). Rendering of spherical light fields. In: *PG '97: Proceedings of the 5th Pacific Conference on Computer Graphics and Applications*, p. 59. IEEE Computer Society, Washington, DC, USA. ISBN 0-8186-8028-8. (Cited on page 44.)
- Jacobs, F., Sundermann, E., De Sutter, B., Christiaens, M. and Lemahieu, I. (1998 3). A fast algorithm to calculate the exact radiological path through a pixel or voxel space. *Journal of computing and information technology*, vol. 6, no. 1, pp. 89–94. (Cited on pages 29 and 93.)

- Lacroute, P. (1995). Fast volume rendering using a shear-warp factorization of the viewing transformation. Tech. Rep. CSL-TR-95-678, Stanford, CA, USA. (Cited on page 42.)
- Lacroute, P. and Levoy, M. (1994). Fast volume rendering using a shear-warp factorization of the viewing transformation. *Computer Graphics*, vol. 28, no. Annual Conference Series, pp. 451–458.
Available at: <http://citeseer.ist.psu.edu/lacroute94fast.html> (Cited on pages 13, 19, 22, and 24.)
- Levoy, M. (2006). Light fields and computational imaging. *IEEE Computer*, vol. 39, no. 8, pp. 46–55. (Cited on page 43.)
- Levoy, M. and Hanrahan, P. (1996). Light field rendering. In: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 31–42. ACM Press, New York, NY, USA. ISBN 0-89791-746-4. (Cited on pages 18, 43, and 48.)
- Meißner, M., Huang, J., Bartz, D., Mueller, K. and Crawfis, R. (2000). A practical evaluation of popular volume rendering algorithms. In: *VVS '00: Proceedings of the 2000 IEEE symposium on Volume visualization*, pp. 81–90. ACM, New York, NY, USA. ISBN 1-58113-308-1. (Cited on page 42.)
- OpenMP Architecture Review Board (2005). OpenMP Application Program Interface 2.5.
Available at: <http://www.openmp.org/mp-documents/spec25.pdf> (Cited on page 25.)
- Pluim, J.P.W., Maintz, J.B.A. and Viergever, M.A. (2003). Mutual-information-based registration of medical images: a survey. *IEEE Transactions on Medical Imaging*, vol. 22, no. 8, pp. 986–1004.
Available at: <http://dx.doi.org/10.1109/TMI.2003.815867> (Cited on pages 55 and 56.)
- Porter, T. and Duff, T. (1984). Compositing digital images. In: *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pp. 253–259. ACM Press, New York, NY, USA. ISBN 0-89791-138-5. (Cited on page 21.)
- Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992). *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA. ISBN 0-521-43108-5. (Cited on pages 57, 58, and 59.)
- Russakoff, D.B., Rohlfing, T., Mori, K., Rueckert, D., Ho, A., Adler, J. and Maurer, C. (2005). Fast generation of digitally reconstructed radiographs using attenuation fields with application to 2D-3D image registration. *IEEE Transactions on Medical Imaging*, vol. 24, no. 11, pp. 1441–1454. (Cited on page 66.)
- Russakoff, D.B., Rohlfing, T., Rueckert, D., Shahidi, R., Kim, D. and Maurer, Jr., C.R. (2003 May). Fast calculation of digitally reconstructed radiographs using light fields. In: *Medical Imaging 2003: Image Processing. Edited by Sonka, Milan; Fitzpatrick, J. Michael. Proceedings of the SPIE.*, vol. 5032, pp. 684–695. (Cited on pages 5, 19, and 48.)

- Sarrut, D. and Clippe, S. (2003 December). Fast DRR generation for intensity-based 2D/3D image registration in radiotherapy. Tech. Rep. RR-LIRIS-2003-002, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon.
Available at: <http://liris.cnrs.fr/publis/?id=1813> (Cited on page 4.)
- Siddon, R.L. (1985 March). Fast calculation of the exact radiological path for a three-dimensional CT array. *Medical Physics*, vol. 12, no. 2, pp. 252–255. (Cited on page 13.)
- van der Bijl, L. (2006). *Verification of patient position for proton therapy using Portal X-Rays and Digitally Reconstructed Radiographs*. Master's thesis, University of Stellenbosch. (Cited on pages iii, v, 4, 6, 7, 9, 55, 71, 72, 85, and 93.)
- Wang, F., Davis, T.E. and Vemuri, B.C. (2002). Real-time DRR generation using cylindrical harmonics. In: *MICCAI '02: Proceedings of the 5th International Conference on Medical Image Computing and Computer-Assisted Intervention-Part II*, pp. 671–678. Springer-Verlag, London, UK. ISBN 3-540-44225-1. (Cited on page 23.)
- Weese, J., Goecke, R., Penney, G.P., Desmedt, P., Buzug, T.M. and Schumann, H. (1999 May). Fast voxel-based 2D/3D registration algorithm using a volume rendering method based on the shear-warp factorization. In: *Medical Imaging 1999: Image Processing. Edited by Hanson, Kenneth. Proceedings of the SPIE.*, vol. 3661, pp. 802–810. (Cited on page 23.)
- Wikipedia (2008 June). Steradian.
Available at: <http://en.wikipedia.org/wiki/Steradian> (Cited on page 98.)
- Zhao, H. and Reader, A.J. (2003). Fast ray-tracing technique to calculate line integral paths in voxel arrays. In: *Nuclear Science Symposium Conference Record, 2003 IEEE*, vol. 4, pp. 2808–2812. ISBN 0-7803-8257-9. ISSN 1082-3654. (Cited on page 33.)