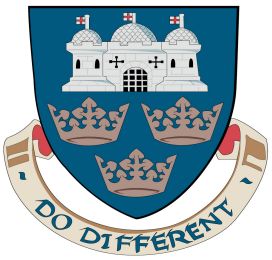
High Speed 2D/3D Medical Image Registration

Osama Mohammad Ahmad Dorgham

School of Computing Sciences
University of East Anglia



November 2010

©This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with the author and that no quotation from the thesis, nor any information derived therefrom, may be published without the author's prior written consent.

To my parents, Mohammad and Haya.

Abstract

In 2D/3D image registration, preoperative 3D CT images are registered with intra-operative 2D X-ray images obtained by fluoroscopy or Electronic Portal Imaging Devices (EPID). These are compared with Digitally Reconstructed Radiograph (DRR) images rendered from CT volumetric data and 2D/3D image registration is established. This process is useful in many medical procedures where surgical planning is undertaken using volumetric data prior to treatment, such as in radiation therapy treatment and image guide surgery.

This thesis examines the problem of accelerating 2D/3D image registration. In practice, we developed methods of accelerating the underlying algorithms (mainly DRR rendering) used in the registration process, at the same time, while checking the performance and accuracy are within an acceptable range. To accelerate DRR rendering we investigated a wide range of computer graphics and image processing techniques using both software (typically executed on the CPU) and hardware (GPU) systems.

A particular focus of this work is 2D/3D image registration prior to radiation therapy treatment and so the algorithms are tested using digital X-ray images acquired at MV and kV energies at the Colney Oncology Centre, Norfolk and Norwich University Hospital (NNUH) using their Varian Linear Accelerator (LINAC) radiation therapy system. Finally, results of the different approaches we developed for accelerating DRR rendering show acceptable accuracy within a 2D/3D image

registration framework. Our results show that we are among the fastest to render DRR images (i.e. we are able to render DRR image from $256 \times 256 \times 133$ CT volume in $\sim$ 24 ms using an NVidia GeForce 8800 GTX and in $\sim$ 2 ms using NVidia GeForce GTX 580), and consequently, speed up 2D/3D image registration without significantly degrading levels of accuracy.

Acknowledgements

In the first place I would like to express my thanks to my supervisors Dr. Mark Fisher and Dr. Stephen Laycock for their advice, guidance, motivation, enthusiasm, and encouragements during my PhD time and research. Dear Mark you were more than a supervisor to me, through you I knew the history, culture, people and traditions of England. Special thanks for helping me getting the skills to plan well and lead my research. I could not have imagined having a better adviser and mentor for my PhD study. Honestly, every time I hear how much some of the PhD students suffer with their supervisors, I realise how much I was lucky to be under your supervision. I can never forget the time we spent discussing our research and traveled to attend the conferences. Dear Stephen, I am really lucky to be under your supervision as well, you were there to help me with full enthusiasm and up to date knowledge. I would like to extend my acknowledgements to my examiners Prof. Anday Day and Prof. Reyer Zwiggelaar (Aberystwyth University).

I owe many thanks to my lab mates Chris Watkins, Jacob Newman, Ibrahim Almajai, Sarah Hilder and Omar Caballero, I will not forget the nice time we had together. Many thanks to the recent lab mates Luke Davis, Oliver Krikland, Khaled Alotaibi and Philip Harding, we really had good time playing squash. Special thanks to Luke for his help of proof reading part of this thesis. There are a lot of names in my mind I would like to mention them, where they were real friends of mine (Jason-Yu Liu, Fabiola Lopez-Gomez, Umar Farooq, Javier

Vzquez Corral and my best friend Paulina Glowacka) and hopefully we will be in touch even after we finish our study and return to our countries. My thanks also goes to Greenman walking club-Norwich (Daniela Furini, Susana Sg, Toni Fernandez, Roberto Montagna, Aurora Perla, Hiroko Furukawa, Francesc Tous) as we had a lot of fun during our walks in Norfolk and the surrounding areas.

I will not forget my friends from my homeland (Anas Trawneh, Talal Shihabi, Lorina Bisharat, Rawan Ibrahim, Ali A. Massadeh, Sara Hazim and Mohammad Ta'amnha) whom I meet here in the UK and shared with them some of my studying time. Big thanks to my friends from Jordan (Ibrahim Nazzal, Mahmoud Abu Atwan, Muthana Hiast, Saed Nawayseh, Mohammad Hiary, Mohammad Arabiat and Osama Hadidi) for their support, I can never forget the smiles on your faces during my visit to Jordan.

I would also acknowledge Alison Vinall from Norfolk and Norwich University Hospital for her support and providing me the data I needed in all the stages of my research.

Finally, all the appreciation and the most special thanks goes to my parents and my brothers for their endless support and encouragement, without them I will not even start to be able to finish. My mum, the support I got from you is not describable; how many times I called you tired from being a way and you cured me by your magical words. My Dad, how many times you helped me to find my way by your wisdom and cleaver ideas. My brothers, I am so lucky by you, you are the best gift for me, you are the ones who can help me, keep me up and enshala we will be friends for ever.

Contents

List of Figures

List of Tables

Publications

The following are publications related to this work by the author:

- O. Dorgham, M. Fisher, S.D. Laycock, A.J. Vinall and W. Holmes-Smith. Fast 2D/3D Image Registration using Accelerated Generation of Sparsely Rendered Digitally Reconstructed Radiographs. *International Journal of Computer Assisted Radiology and Surgery*, Volume 5, Supplement 1, pp. S68, June 2010.

- Osama Dorgham, Mark Fisher and Stephen Laycock. Performance of a 2D-3D Image Registration System using (Lossy) Compressed X-ray CT. *The Annals of the BMVA*, Volume 3, pages 1–11, 2009.

- Osama Dorgham, Mark Fisher and Stephen Laycock. Accelerated Generation of Digitally Reconstructed Radiographs using Parallel Processing. *In Proc. Medical Image Understanding and Analysis 2009*, Jul. 14-15 2009, pages 239-243.

- Osama Dorgham, Mark Fisher and Stephen Laycock. Improving the Performance of 2D/3D Medical Image Registration using Lossy Compressed Data. *In Proc. IPEM Managing Motion in Radiotherapy Meeting*, Manchester Dental Education Centre, UK, 29 April 2009.

- O. Dorgham, M. Fisher and S. Laycock. Accelerated generation of digitally reconstructed radiographs. *In Proceedings of the 5th UKRI PG Conference in Biomedical Engineering and Medical Physics 2009*, University of Oxfored, UK, 2009, pages 19-20.

- O. Dorgham, M. Fisher and S. Laycock. Performance of 2D/3D Medical Image Registration using Compressed Volumetric Data. *In Proc. Medical Image Understanding and Analysis 2008*, Jul. 2-3 2008, pages 261-265.

Awards and Prizes

- 1^{st} prize, present around the world (PATW), Title: Fast Generation of Digitally Reconstructed Radiographs for 2D/3D Medical Image Registration, Institution of Engineering and Technology (IET), 18th March 2009.
- Best presentation, CMP postgraduate research day, Title: Fast 2D/3D Image Registration using Accelerated Generation of Digitally Reconstructed Radiographs with Automatic Selection of Region of Interest, University of East Anglia, 9th October 2009.

List of Abbreviations

2D	-	Two Dimensional
3D	-	Three Dimensional
AEC	-	Automatic Exposure Control
AF	-	Attenuation Fields
ASD	-	Amorphous Silicon Detector
CPU	-	Central Processing Unit
CT	-	Computed Tomography
CUDA	-	Compute Unified Device Architecture
DOF	-	Degrees Of Freedom
DRR	-	Digitally Reconstructed Radiograph
EBCT	-	Electron Beam Computerised Tomography
EPID	-	Electronic Portal Imaging Device
FR-DRR	-	Full Resolution DRR images
FS-DRR	-	Full Size DRR images
GFLOPS	-	Giga Floating point Operations Per Second
GPGPU	-	General Purpose computing on Graphics Processing Unit
GPU	-	Graphical Processing Unit
HU	-	Hounsfield Units
IR	-	Infrared
kV	-	Kilo electron-Volt
KVCT	-	Kilo Voltage Computed Tomography
LINAC	-	Linear Accelerator
LNC	-	Local Normalised Correlation
LUT	-	Look-Up-Table
MI	-	Mutual Information
MIMD	-	Multiple Instruction Multiple Data
MLC	-	Multileaf Collimator
MR	-	Magnetic Resonance

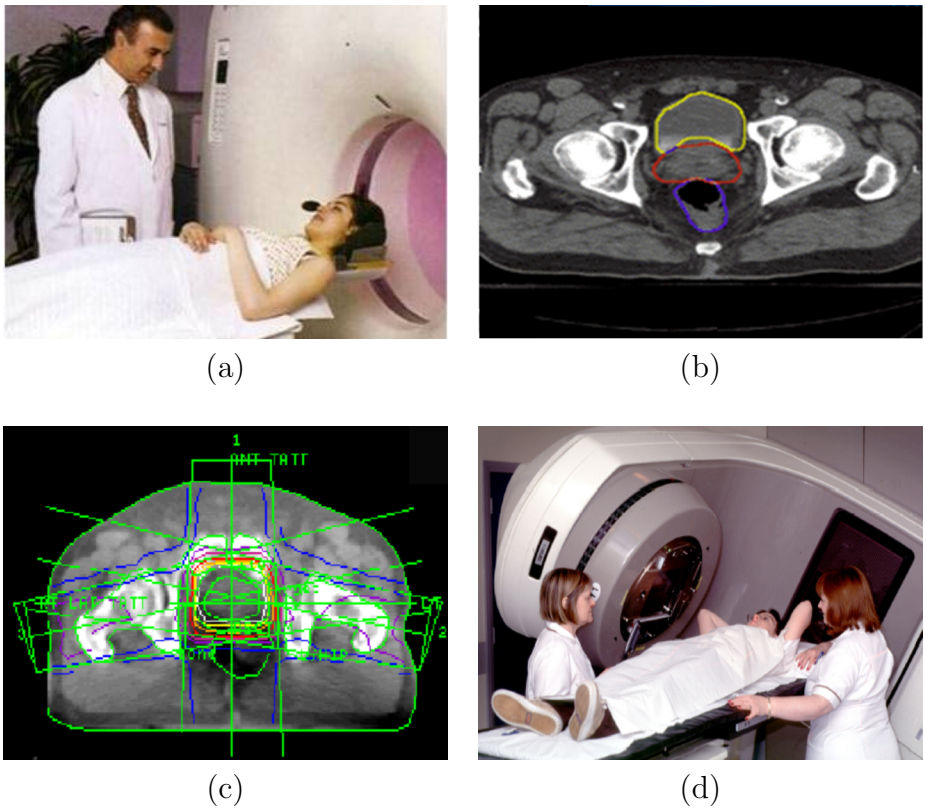
MSE	-	Mean Square Error
MV	-	Mega Electron-Volt
MVCT	-	Mega Voltage Computed Tomography
NCC	-	Normalised Cross Correlation
NNUH	-	Norwich and Norfolk University Hospital
PAF	-	Progressive Attenuation Field
PET	-	Positron Emission Tomography
PSNR	-	Peak Signal-to-Noise Ratio
ROI	-	Regions Of Interest
RR-DRR	-	Reduced Resolution DRR images
SIMD	-	Single Instruction Multiple Data
SNC	-	Sparse Normalised Correlation
SP-DRR	-	Sparsely DRR images
SPECT	-	Single-Photon Emission Computed Tomography
SVM	-	Support Vector Machine
T	-	Transformation
TPS	-	Treatment Planning System
TRE	-	Target Registration Error

Chapter 1

Introduction

Radiation therapy is an effective treatment for many types of cancer and as external beam therapy is relatively easily delivered, it remains a widely used treatment [154]. The radiation therapy process involves multiple steps as illustrated in Figure 1.1. The process begins with data acquisition as a CT volume (Figure 1.1 (a)), then tumour localisation followed by treatment planning (Figure 1.1 (b) and (c)) and finally the radiation delivery (Figure 1.1 (d)). Radiation treatment is delivered in fractions, each lasting 15-30 minutes and the whole process, from beginning to end, takes 4-6 weeks [79][31].

Since radiation damages both healthy and malignant cells, it needs to be accurately targeted. Accurate positioning of the patient prior to (or during) treatment is a vital component in achieving a successful outcome and one that becomes increasingly important when Intensity Modulation Radiotherapy Treatment (IMRT) is used [130]. 2D/3D medical image registration, an image guided procedure [187] used to match preoperative images and plans to images captured intraoperatively, is used to align the patient prior (or during) to the delivery of radiation therapy treatment [104]. It returns a rigid transformation (rotation and translation) which provides parameters that are used to position the couch, thereby aligning the patient's anatomy with that used for the planning process [11]. Digitally reconstructed radiographs (DRR) images are rendered from CT volumetric data by summing the attenuation of each voxel along known ray paths through the CT volume. Generating projection radiographs is computationally more demanding than



(a) (b)

(c) (d)

Figure 1.1: The main steps of radiation therapy treatment: (a) data acquisition (i.e. CT scan), (b) tumour localisation, (c) treatment planning and (d) radiation therapy delivery [169][71][108][149].

3D surface rendering as potentially all the voxels contribute to the process. However, recent research in automatic 2D/3D registration and online motion tracking [22] requires efficient registration of many DRRs (e.g. Lung cancer registration process should be faster than 0.33 Hz as the breath rate for adult range between 12-20 per minute [170][91]) and this has motivated research into fast algorithms and hardware acceleration [90][49][124][86]. In [49] Göcke et al. comprehensively review and compare few volume rendering approaches, and discuss a number of specific optimisations, concluding that shear-warp factorisation (i.e. an object space based volume rendering algorithm, more details about shear-warp factorisation are presented in Section 3.4.3) [89] with runlength encoding is roughly a factor of six times faster than a direct approach and that the registration accu-

racy is unaffected by the rendering approach so long as an appropriate resolution is chosen. Russakoff et al. [141] implemented a special ray-based data structure called an Attenuation Field (AF) to be used in the generation of a DRR instead of the conventional ray casting method. (more detailed information is presented in Section 3.5). Other work by Penney et al. [124] and Knaan and Joskowicz [86] compare approaches for 2D/3D image based registration, both reporting similar surface Target Registration Errors (TRE) of the order of 1-2 mm. Hardware acceleration is the focus of other work by Ruijters et al. [140], Gross [54], Wang [178], Bethune et al. [13], Spörk [160], Mori et al. [111] and Birkfellner [186]. The latter uses a technique known as splatting which has been developed to efficiently render irregular meshes, such as those generated by Octrees, using the Graphics Processor Unit (GPU).

1.1 Motivation

The motivation for improving the speed of 2D/3D image registration can be explained by the following reasons:

Firstly, patient positioning is an important procedure for the effective and safe delivery of radiation therapy treatment that can be solved using 2D/3D image registration. An accurate and fast procedure for patient positioning saves time and reduces errors in targeting malignant cells at the tumour site. A lot of effort and time is spent by doctors and physicist in the early stages of diagnoses and treatment planning and this time can be wasted due to inaccurate positioning or worse lead to harmful side-effects. Studies have shown that "Efficiency of the radiation therapy planning depends on the patient setup at each daily fraction" (i.e. daily fraction: each fraction of the treatment process) [84]. Fast procedures for patient positioning lead to the treatment of more patients in the same slot of time. We focused on improving (accelerating) DRR rendering as this process is computationally expensive and the most demanding component of the registration method. Additionally, new image guided radiotherapy techniques will benefit from fast image registration. Recently, Su et al. [163] described a method for organ motion tracking which depends on fast rendering DRR images and other researchers have clinically demonstrated real-time tracking and treatment delivery

systems [29][38][63]. Applications of such systems are limited by latencies within the registration loop.

1.2 Research Approach

The first step of this study starts by analysing the 2D/3D image registration scheme in order to identify potential bottlenecks within the process and to understand interrelations between its components. Then we investigate different methods of accelerating the rendering of DRR images such as:

1. Data sampling and CT volume compression, including Octree CT volume compression methods, sampling by reducing the number rays and sampling by reducing the number of intersection points inside the CT volume.

2. Parallel processing, this includes two methods using the software CPU and hardware GPU.

3. Sparsely rendered DRR images which uses entropy measure for image blocks selection.

Concurrently with the improvement for the image registration components we evaluate the performance of the process by checking if the improvements made affect the 2D/3D registration performance and if this remains in an acceptable range for clinical radiation therapy use.

1.3 List of Contributions

This thesis makes the following contributions:

- It proposes a 2D/3D image registration system using compressed CT volumetric data (Octree compression) and shows that such a system is able to achieve clinically acceptable results even when the CT volume is subject to high levels of compression.

- It develops parallelised software (i.e. CPU - OpenMP) implementation of a DRR rendering method and demonstrates this is capable of achieving accelerated 2D/3D image registration.
- It also develops a parallelised hardware (i.e. GPU - CUDA) implementation of fast DRR rendering method and demonstrates its acceleration to 2D/3D image registration.
- It investigates a method of performing 2D/3D image registration using reduced resolution DRR images and reports on its accuracy.
- Also, it investigates a method of performing 2D/3D image registration using sparsely rendered DRR images and reports on its accuracy.
- It integrates the DRR rendering algorithm within a 2D/3D registration framework that includes a novel optimisation approach which requires fewer iterations than the standard hill climbing method.

1.4 Thesis Organisation

The remainder of this thesis is organised in the following manner. Chapter 2 describes different radiological image modalities and intensity modulated radiation therapy systems used in image registration. Chapter 3 examines 2D/3D image registration and all its components with detailed description for the methods that we implemented in our registration system. Chapter 4 examines the performance of 2D/3D image registration for using the compressed CT volumes to accelerate the rendering of DRR images. Chapter 5 examines a way of accelerating the rendering of DRR images by parallelising the casting of rays. Chapter 6 introduces a way of accelerating the 2D/3D image registration by developing a hybrid system combined the CPU and GPU. Chapter 7 examines a way of accelerating the rendering of DRR images by only rendering a fragment of the DRR images with performance evaluation for the 2D/3D image registration using this method of accelerating the DRR rendering. Chapter 8 concludes this study and discusses the possibilities of future work.

Chapter 2

Radiology and Radiation Therapy

The scope of this chapter is limited to an overview of different medical image modalities and radiation therapy systems that are related to our research (i.e. modalities and systems that have been used or could be enhanced with the addition of new methods described in this thesis, to achieve improved performance). Section 2.1, provides an introduction to X-ray generation; we are particularly interested in two X-ray energies, kV and MV, used in radiotherapy. We then describe the interaction between X-rays and materials and their effect on the quality of kV and MV images. Section 2.2 describes computed tomography and its technological development. In Section 2.3 we briefly review other imaging modalities such as magnetic resonance imaging and photon emission tomography. Finally, in Section 2.4 different systems for radiation therapy are described as this work provides the motivation for improvements in the 2D/3D registration process which is a particular focus of this research.

2.1 X-ray Imaging

X-ray imaging has been used in various medical applications for over 100 years. In 1895, Röntgen published his initial results of using X-rays based on his observation of fluorescence [75][139]. Figure 2.1 shows the first published X-ray image.

Table 2.1 shows a timeline of medical imaging. X-ray imaging was the first modality used in medical applications and has been used to develop other med-

Figure 2.1: An X-ray image of Anna Berthe Röntgen's hand, taken on 22-12-1895 [3].

ical imaging techniques (e.g computed tomography). The main reason for the widespread use of X-rays in medical and dental applications is firstly, that they are simple to generate and detect. Secondly, it is still the cheapest modality for obtaining medical images [185].

2.1.1 X-ray Generation

X-rays are high energy photons generated when electrons strike a metal target. The electrons are emitted from a heated filament or cathode and accelerated to collide with the metal target (anode). X-rays will be generated as a result of this collision. A schematic drawing for an X-ray tube is shown in Figure 2.2.

X-rays are generated due to several processes which occur when the accelerated electron penetrates the anode surface. Firstly, X-rays are produced when electrons are diffracted and slowed down by the Coulomb fields of the atoms in the anode material. Secondly, due to the deceleration that results from the interaction with the orbital electrons and the atomic nucleus. These decelerations produce so called Bremsstrahlung radiation. Occasionally, the entire energy of the accelerated electron is transfered into a single photon [21]. Thirdly, the accelerated electron could

1895	The first **X-ray** picture has been taken by the German physicist Wilhelm Conrad Röntgen, showing the skeletal composition of his wife's left hand [120].
1950	First **Positron Imaging** Device [120].
1952	First **Clinical Positron Imaging** Device [120].
1954	First reports of cardiac **Ultrasound Imaging** have been reported by Elder and Hertz [82].
1955	First **Fluoroscopic movies** that allowed dynamic X-ray imaging of moving scenes. Providing a new information of the beating heart and its blood vessels [70].
1972	**Computed Tomography (CT)** scanning machine has been invented by the British engineer Godfrey Hounsfield of EMI Laboratories, England, and the South African born physicist Allan Cormack of Tufts University, Massachusetts [152][70].
1973	The first **Magnetic Resonance Image (MRI)** has been developed by the American Paul Lauterbur and the English Mansfield of Thorn-EMI laboratories [120].
1974	The first **Positron Emission Tomography (PET) camera** has been developed by the American Michael [120].
1980	The first clinical **MRI** of the brain was first done on a patient [70].
1985	The first **Clinical PET** scanning has been developed by scientists at the University of California [70].
1989	The first **Spiral CT** scanning has been developed which allowed fast volume scanning of an entire organ during a single, short patient breath hold of 20 to 30 seconds [70].
2000	The first **PET/CT scanner** is invented and named by the Time Magazine to be the medical invention of the year [120].

Table 2.1: Medical imaging timeline.

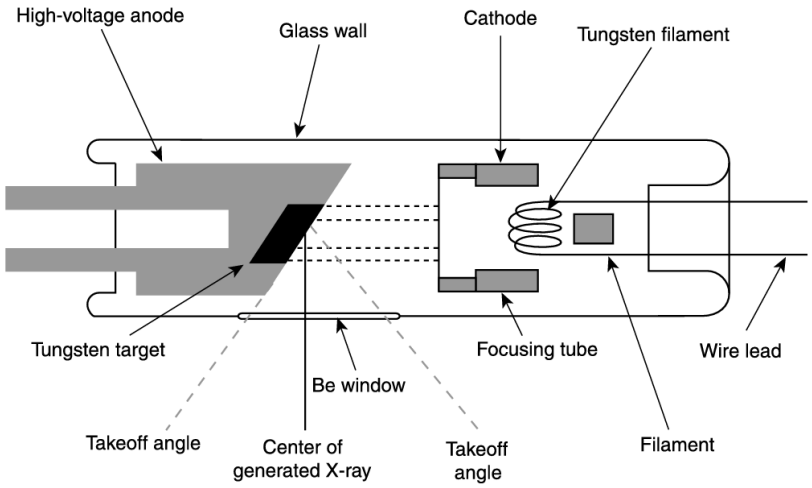


Figure 2.2: Schematic drawing for the Xray tube [73].

knock out an electron close to the nucleus in the metal atom which will be filled by an electron further out from the nucleus. This process will cause a difference in the binding energy which will result in the emission of a monoenergetic photon. In 1917, Barkla was awarded a Nobel Prize for this discovery and research that also proved that X-rays are electromagnetic waves, with a range of wavelengths roughly between 10^{-8} m and 10^{-13} m [21][1], a range invisible to the human eye, as shown in Figure 2.3.

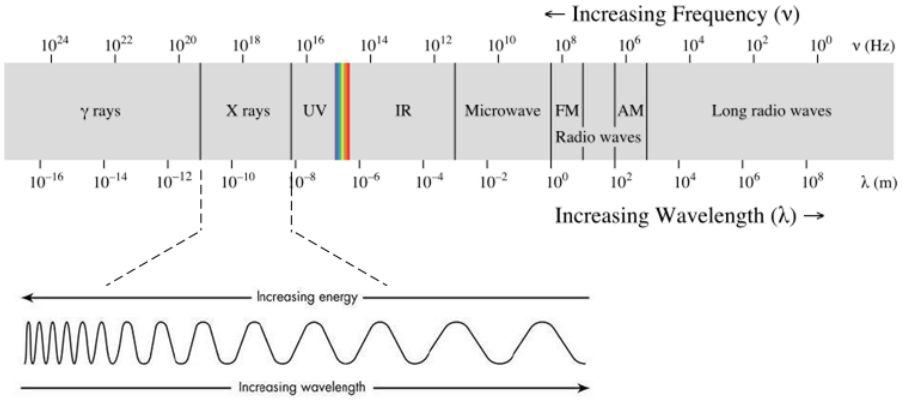


Figure 2.3: X-rays in the electromagnetic spectrum [138].

X-rays used in medical diagnostics are produced by acceleration voltages chosen

between 25 kV and 150 kV [21], while those used in radiation therapy employ voltages between 6 MV and 20 MV [30]. In our experiments (Chapter 7) X-rays used for kV diagnostic imaging range from (79-81) kV and in radiation therapy (portal imaging) a 6 MV source is used.

2.1.2 X-ray Material Interaction

X-ray interactions with matter is important in medical imaging and diagnostic applications. The interaction (absorption) of X-ray photons by the human body (e.g, bone, muscle tissue, epithelial cells, etc.) is used to generate medical images. The interaction of high energy ionising X-rays with the human body is used in radiation therapy treatment to destroy cellular DNA. X-ray interaction with the human body results in one of three cases. Firstly, the X-ray could penetrate the section of body without interaction (no loss in energy). Secondly, it could be totally absorbed by the body materials (total loss in the photon energy). Thirdly, it could be deflected from its original direction or scattered (partially loss in photon energy) [161]. These cases are illustrated in Figure 2.4.

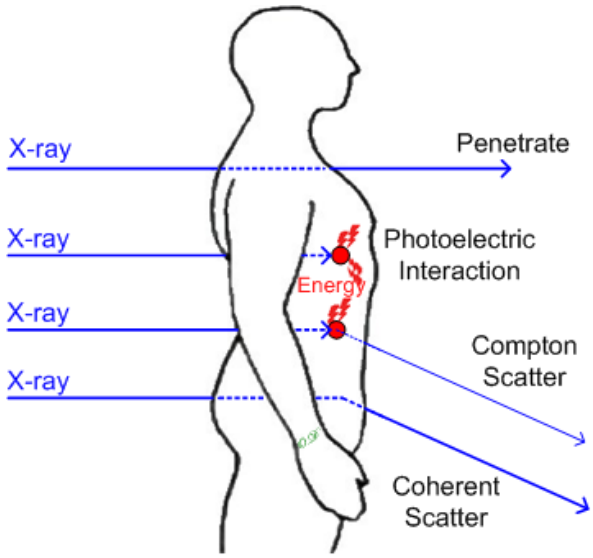


Figure 2.4: Illustration for the cases of X-ray interaction with human body materials [161].

In medical image applications, the effects of a photons energy resulting from

interactions with the human body materials are explained by the photoelectric effect, Compton scatter and coherent scatter. In the photoelectric effect, the energy of the X-ray photon is absorbed by an orbital electron which may be ejected or experience a change in orbit (energy level), producing characteristic radiation. In Compton scatter, some of the X-ray photon energy is absorbed by an electron, while the X-ray photon travels in another direction with less energy. In coherent scatter, no X-ray photon energy will be deposited in the material, it is pure scattering interaction as illustrated in Figure 2.4.

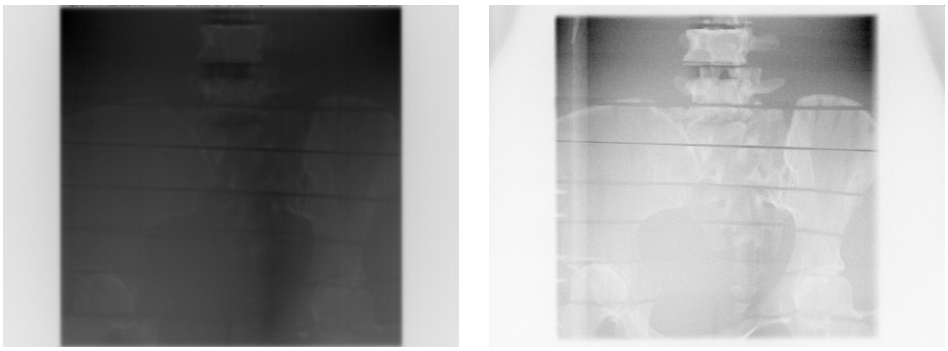
X-ray imaging for medical applications relies on their attenuation due to different materials within the body. Since different materials will give different levels of X-ray intensity attenuation, then a projection image can be generated. Levels of attenuation are governed by the Beers-Lambert law [83] which can be expressed as follows:

$$I(\eta) = I_0 * exp^{-\mu(\eta)}$$

Where I is the final X-ray intensity, I_0 is the initial X-ray intensity at the source, μ is the linear attenuation coefficient for the material through which the X-ray is cast and η is the length of the X-ray path. As there is quite a big difference between the attenuation levels of soft tissue and bone (at kV energy) X-ray images with good contrast can be effectively captured for different medical applications [75]. In our application of 2D/3D image registration we collected two types of X-ray images which differ from each other by the X-ray source energy levels; MV and kV. The significant point relating to the X-ray energy is that the probability of photoelectric interaction occurring within a given material drops sharply as the X-ray energy is increased [161]. This explains the relatively poor quality of MV X-ray images compared to the quality of kV X-ray images, as shown in Figure 2.5. Further details about the effect of using MV and kV X-ray images in the 2D/3D registration are presented in Chapter 7.

2.1.3 X-ray Detection

X-ray detection is a way of transferring the corresponding photon energy into an electrical signal. There are two methods of X-ray detection; direct and in-



(a) (b)

Figure 2.5: Example of MV and kV X-ray images that shows the effect of using different levels of X-ray energy during the process of generating X-ray images, where the image (a) generated using 6 MV energy of X-ray and (b) generated using 80 kV energy of X-ray.

direct. The indirect method uses special materials called Scintillators that are able to convert X-rays into visible light which are later detected by conventional photodetectors, as illustrated in Figure 2.6 (a). This method of X-ray detection provides good quality analog images but this type of image is incompatible with modern digital storage. A modification of the conventional method uses a fluoroscopy screen and camera to produce analog images directly to a TV screen, as illustrated in Figure 2.6 (b). The modified method suffers from multiple conversion steps (i.e. X-rays to electrons to light, then to the camera) which leads to poor image quality. A direct conversion process solves these problems by using a semiconductor detector plane to convert X-rays directly into electrical signals as illustrated in Figure 2.6 (c). This method of capturing digital images leads to the generation of good quality images, that have lower storage cost which makes this method suitable for the needs of modern healthcare [75]. Our 2D/3D image registration algorithms are tested using digital X-ray images acquired at MV and kV energies at the Colney Oncology Centre, Norfolk and Norwich University Hospital (NNUH) using a direct method of X-ray detection.

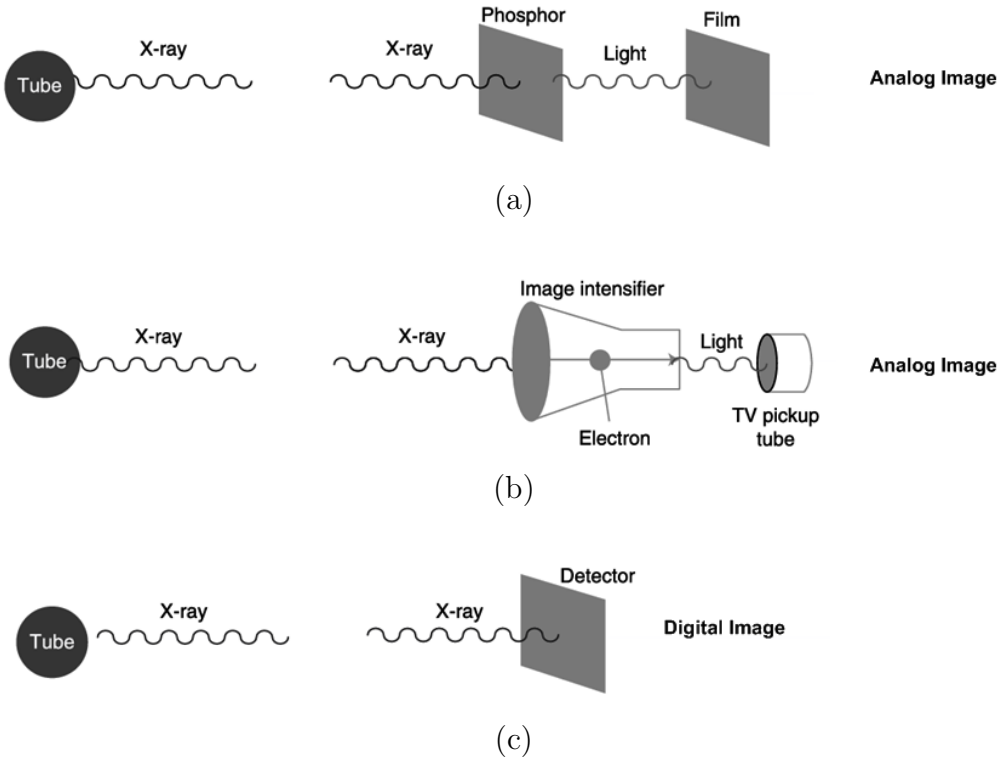


Figure 2.6: Illustration for the different methods of detecting X-ray images. Where (a) shows the conventional method of X-ray image detection, (b) shows fluoroscopy X-ray image detection and (c) shows the digital method of X-ray image detection.

2.2 Computed Tomography

The method of Computed Tomography (CT) imaging was developed in the early 1970's by an English scientist Godfrey Hounsfield who was (jointly) awarded a Nobel Prize in medicine in 1979. CT images are generated using X-rays to obtain thin slices (images) through the patient's body. CT images provide more detail (i.e. higher contrast) for soft tissue organs (i.e. liver, kidney, muscles, etc.) compared to conventional X-ray radiography as the computed tomography yields X-ray attenuation per unit volume and is not summed as in the case with conventional projection imaging. The basic idea behind CT imaging is to reconstruct a two dimensional image slice using one dimensional X-ray projections that are acquired from different angles [179]. Consequently, a sequence of CT slices could be used to reconstruct a three dimensional volume that in turn could be used in different medical applications (e.g. surface rendering, image registration, image segmentation etc.). In 2D/3D image registration we need to render two dimensional projections (i.e. digitally reconstructed radiographs) of the 3D CT volume. During the last four decades, methods of acquiring CT images have been refined in several *generations* of scanners. These mainly differ from each other based on the scanning configuration, scanning motions, detector arrangement and geometry of the X-ray source. Briefly, in the following part of this section we will describe the geometry for each of the generations and discuss its main pros and cons.

2.2.1 First Generation

The first generation is the simplest method of generating CT images. A highly collimated X-ray pencil beam and detector is used to obtain multiple measurements of X-ray transmission, translated in linear motion across the patient's body. Projections are obtained by rotating the X-ray source and detector about the patient's isocentre by approximately 1° for each projection as illustrated in Figure 2.7. The rotation-translation motion is repeated to cover all the sides until the source and detector have rotated through 180°. However, this method requires a long time per each scan (i.e. approximately 5 minutes) [32].

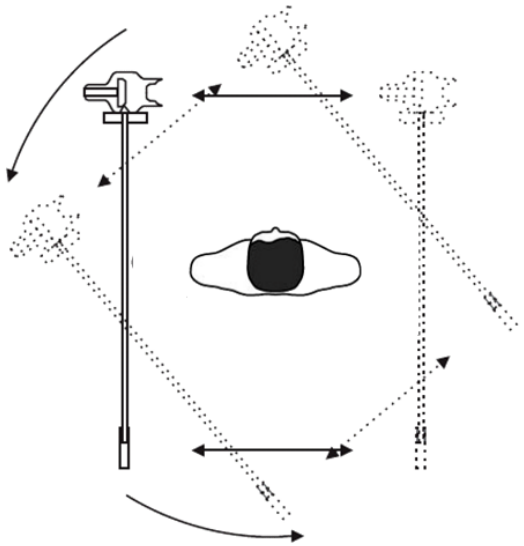


Figure 2.7: First generation of CT [96].

2.2.2 Second Generation

The second generation CT used a narrow fan beam of X-rays and linear detector array as illustrated in Figure 2.8. A translation and rotation motion of the X-ray source and detectors were still employed, as in the first generation but a 10° step in rotation angle was now possible due to the use of a fan beam. This reduced the time need to acquire a scan to 10-90 seconds depending on the manufacturer, but increased the computational complexity of the reconstruction [21][32].

2.2.3 Third Generation

Third generation machines used a wide fan beam of X-rays with a large number of detector elements that are able to cover the patient completely so the translation motion of the previous designs was avoided. Moreover, the X-ray source and the detectors are mechanically coupled together and both rotated together around the isocentre as illustrated in Figure 2.9. The main aim of this design is to reduce the acquisition time to less than 20 seconds. In order to reduce errors due to motion artefacts the patient was able to hold their breath for this short time [21][32].

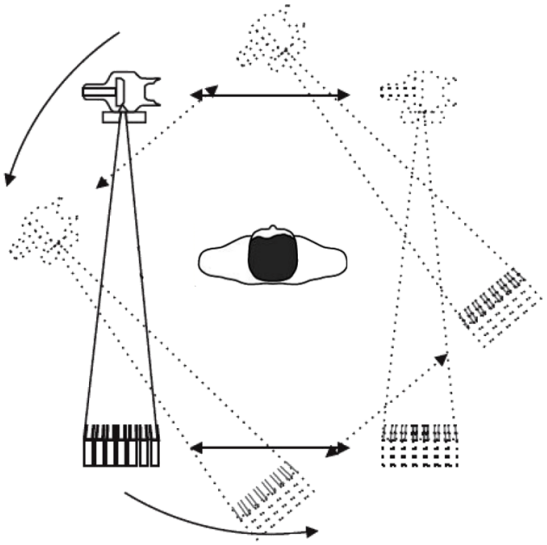


Figure 2.8: Second generation of CT [96].

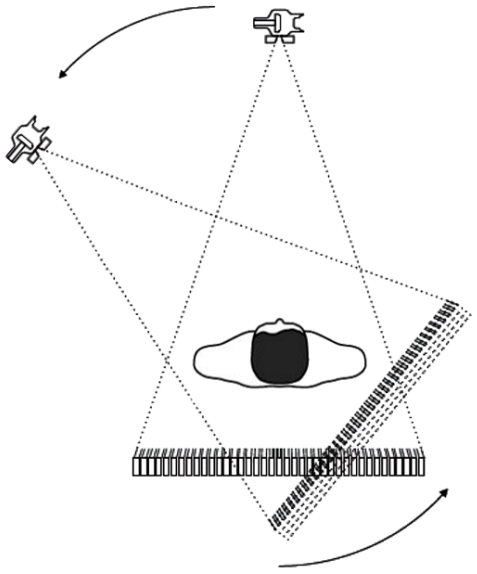


Figure 2.9: Third generation of CT [96].

2.2.4 Fourth Generation

Fourth generation designs used the same wide X-ray beam fan as in the third generation but with a fixed full ring of detectors around the isocentre as illustrated

in Figure 2.10. Acquisition time for the projection data can be performed in less than a second [20], with inerscan delay (elapsed time between two scans) of 100 milliseconds. Interscan delay is avoided by enabling the X-ray tube to move in spiral motion, this brings a significant reduction in the total scanning time [24]. Our research used CT acquired from a fourth generation $GEHiSpeedFX/i$ CT scanner located at NNUH, operating at 120 kV.

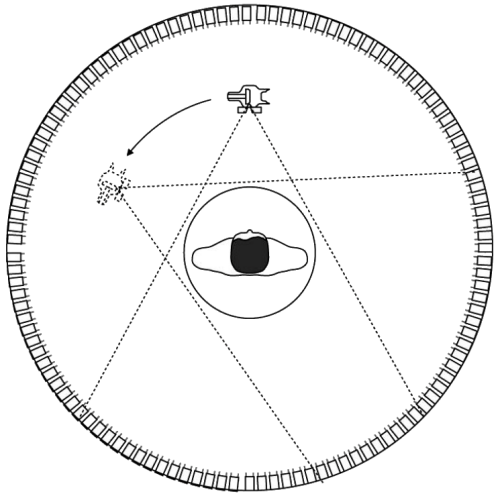


Figure 2.10: Fourth generation of CT [96].

2.2.5 Fifth Generation

The main reason for developing the fifth generation is to reduce slice acquisition time as much as possible, which allows the system to be used in cardiac imaging. Fifth generation CT is also called electron beam computerised tomography (EBCT). Here, an electron source radiates a non localised X-ray beam. The electron beam is focused onto a ring of wolfram targets which are arranged in a half circle shape around the patient that radiate the required X-ray beams for the CT system imaging [21] as illustrated in Figure 2.11. Acquisition time for the projection data can be performed in about 50 ms, which is enough to image a beating heart with acceptably small motion artefacts [16].

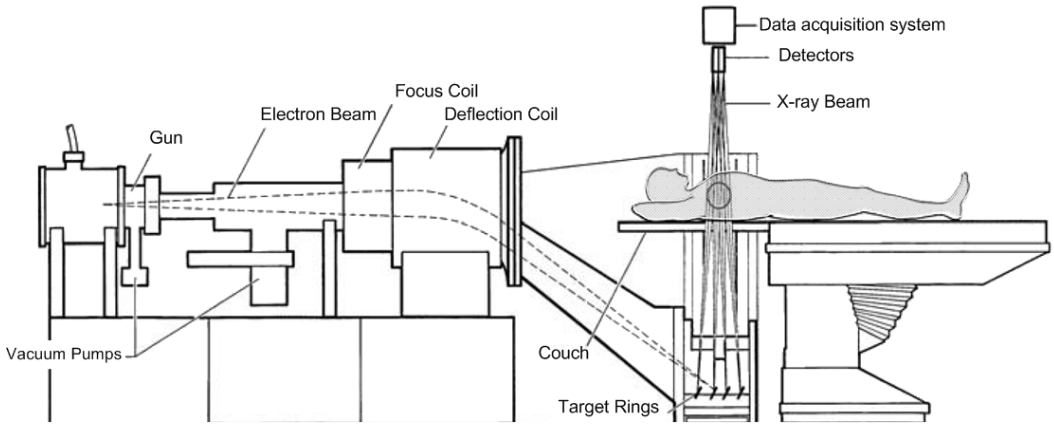


Figure 2.11: Fifth generation of CT [19].

2.3 Other Radiology Imaging Modalities

In this section we briefly highlight other radiology modalities that could be used, together with methods described in later chapters to improve the speed of the 2D/3D image registration.

2.3.1 Magnetic Resonance Imaging

Magnetic Resonance Imaging (MRI) was developed in 1973 by Lauterbur and Mansfield. Images are generated using a nuclear magnetic resource that is able to visualise the internal structure of the human body. Comparing MRI to CT scans, MRI provides better contrast between different body's tissues, this makes it useful for imaging the brain, spinal chord, other internal organs such as lung and liver or even for bones and joints as illustrated in Figure 2.12.

The basic idea behind MRI is that the hydrogen nuclei, which make up 80% of all atoms in the human body are aligned as a nuclear magnetised atoms by powerful magnetic fields. In the next step, radio frequency fields are used to alter the alignment of the previous magnetised atoms. This causes the production of magnetic signals from the hydrogen nuclei, these are detected by the MRI scanner and reconstructed as an MRI image [20][115]. The focus of Chapters 4-7 is in different methods of speeding up the rendering of DRRs. We demonstrate these methods of DRR rendering using X-ray-CT volumes only. However, recent

Figure 2.12: Example of MR image for the human body, illustrating the better contrast of MR over CT imaging [65].

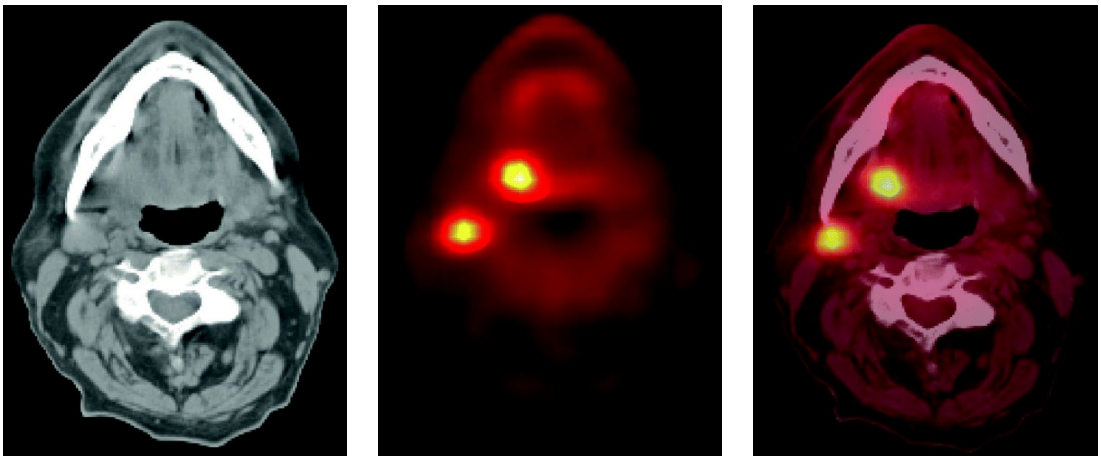
research on rendering DRR images from MR volumes reported by Ramsey et al. [132] uses MR-DRR images in a clinical utility for the setup and verification of patients with intracranial lesions. Similarly, Chen et al. [27] rendered MRI-based DRR images to facilitate initial patient setup process, where the MR-DRR rendering is an essential process in their application of a MRI-based treatment planning system. These applications could be made more efficient using methods for speeding up the rendering process which we discuss in Chapters 4-7.

2.3.2 Positron Emission Tomography

Positron Emission Tomography (PET) is a type of nuclear medicine imaging that uses short-lived radionuclides to visualise body functions (e.g. blood flow, oxygen use and sugar metabolism). This gives doctors a tool to be able to evaluate the functionality of tissues and organs [131][57]. PET images are acquired using gamma-rays emitted by the decay of radio-isotopes bound to molecules [99] as follows. It begins with the injection of a radio-pharamaceutical, then a detector scans the organ or tissues of interest after a specific delay to allow for delivery and uptake. When the radio-isotope starts to decay it emits a positron that travels for a short distance before annihilating with an electron which produces two high-energy (511 kV) photons, that scatter in nearly opposite directions [99]. Summing many events results in quantities that approximate line integrals through the radio-isotope distribution which is then used to reconstruct the PET image (for more details about PET image reconstruction, see [99]). A gamma camera is used to detect emitted photons and a two dimensional histogram of the detected events forms a projection image of the distribution of the radio-isotope [99].

Recently, systems have been built that combine PET and CT scanners to generate PET-CT images. This type of images are generated using two different types of radiation; gamma and X-ray for the PET and CT respectively. The main aim of combining this two imaging modalities is to acquire images that are able to describe the patient's body organs in a functional and structural way as shown in Figure 2.13.

PET-CT images have enabled an improvement to the planning and registration for radiation therapy, this in turn has increased the need to develop new methods for 2D/3D registration [26]. Method of accelerating the 2D/3D registration process are applicable for these volumes too. Moreover, our fast rendering method could be easily applied to allow both projections (DRRs) and surfaces to be rendered through 3D PET-CT volumes using the maximum intensity projection (MIP) method described in [97][85][175][17][122].



(a) (b) (c)

Figure 2.13: An example shows the importance of the functional and structural images together, where (a) shows the CT image component, (b) shows the PET image component and (c) shows the PET-CT fused image [113].

2.4 Radiation Therapy Systems

The common and main goal of radiation therapy systems is to deliver radiation to the tumour site in the correct doses and location without affecting the surrounding healthy tissues. There are several different types of radiation therapy systems such as: Linear accelerator (LINAC), CyberKnife and Tomotherapy. In the following subsections we introduce each of these different radiation therapy systems and its methods of image registration for the patient positioning procedure.

2.4.1 Linear Accelerator (LINAC)

LINAC accelerator is the most commonly used radiation therapy system for patients with cancer. It delivers a high dose of X-ray energy to the region of the tumour. It can be used for a stereotactic radiosurgery, this makes it suitable to treat most areas of the body [121]. The system consists of MV X-ray- source, electronic portable imaging device (EPID), multileaf collimator (MLC) and patient support system (PSS) [180] as illustrated in Figure 2.14.

The MV X-ray source is able to move in a circular motion around the patient,

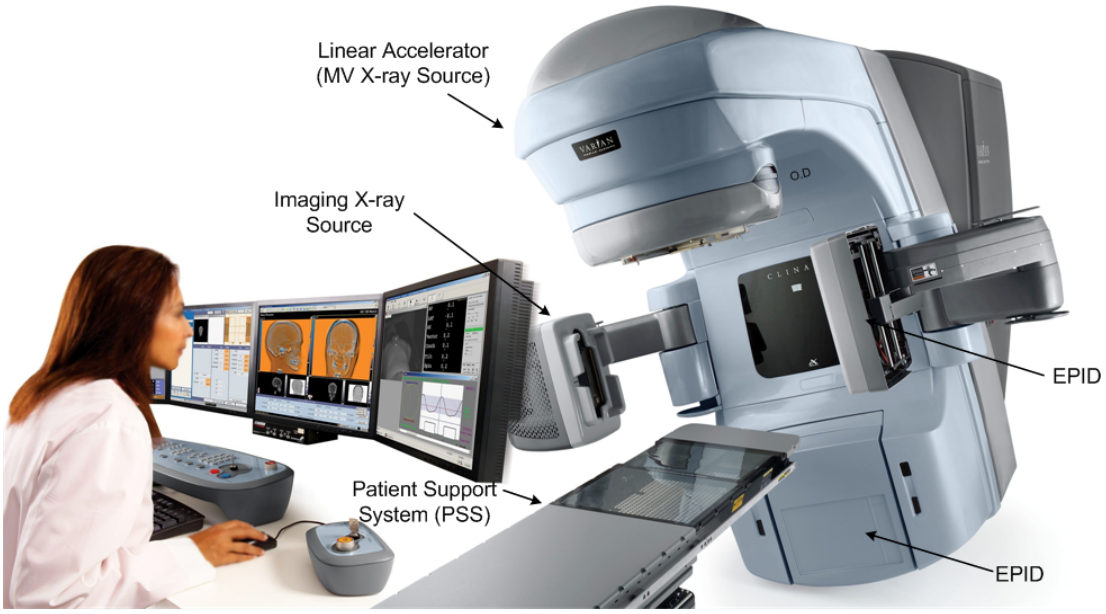


Figure 2.14: A LINAC system showing the major components [164].

while the couch or the PSS can be rotated or translated in six degrees of freedom, enabling the system to reach any point on the patient's body [174]. More technical details about the LINAC system that are relevant for our study are presented in Section 7.4.1. Patient positioning prior radiation therapy treatment is performed using 2D/3D image registration. Our research investigates several different methods for speeding it up and investigates the registration performance in the framework of kV/MV or kV/kV registration problem.

Recently, intensity modulated radiation therapy (IMRT) has become a vital tool in meeting the main goal of radiation therapy systems (i.e. delivering radiation to the tumour site in the correct doses and location without affecting the surrounding healthy tissues) [167]. In IMRT, radiation beams are shaped to closely approximate the shape of tumour, this enables a more precise conformal radiation dose to be delivered to the tumour area [77]. To achieve the goal of IMRT, a multileaf collimator (MLC) is used to turn beams on or off in order to control the intensity and shape of it as shown in Figure 2.15.

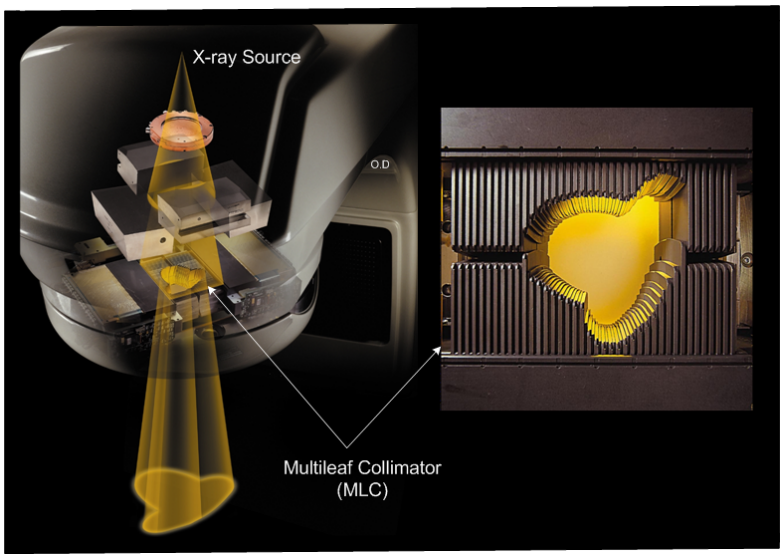


Figure 2.15: The principle of intensity modulated radiation therapy using an MLC [165][164].

2.4.2 Cyberknife

"The CyberKnife (Accuray, Inc., Sunnyvale, CA) is an image-guided frameless radiosurgery system" [48]. There is currently only one machine of this type located in the UK (in a Harley Street Clinic). It is capable of irradiating tumors stereotactically with the tumor location (moving organ) being fed back to a computer controlled robot [180]. The design of Cyberknife enables it to be used in the treatment of cancers located in a variety of body sites (i.e. brain, lung, spine, liver, prostate, pancreas) [137]. The system consists mainly of a robotic controller, an X-ray radiographic locating system and a light weight 6 MV linear accelerator head [29] as shown in Figure 2.16.

The robotic controller is capable of six degrees of freedom movement. Feedback from the computerised localisation system uses two fixed X-ray sources (i.e. 100-120 kV [29]) that enable orthogonal images of the targeted cancerous region to be acquired. The infrared (IR) tracking system is used to continuously synchronise the beam delivery to the tumour's motion which eliminates the need for breath-holding technique [147]. The patient positioning system is also synchronised with the treatment system through the robot which aligns the patient in six degrees of freedom enabling precise and fast patient setup.

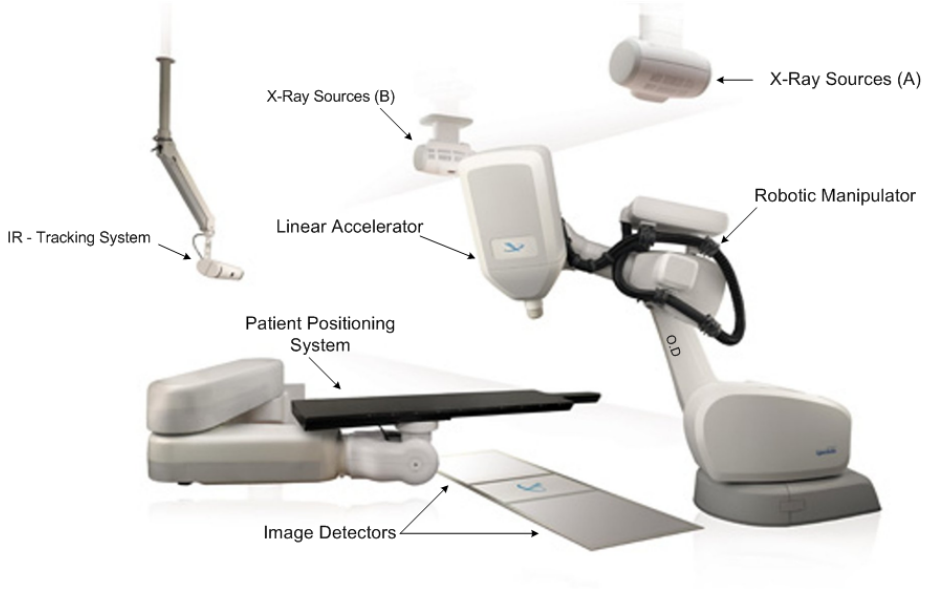


Figure 2.16: A system overview of the Accuray® Cyberknife showing its major components [2].

The target location system (i.e. X-ray sources (A) and (B) and the image detectors) generate images that are fed into image-guidance software which uses 2D/3D image registration to track the targeted tumour by periodically registering these X-ray images to previously generated DRR images [2]. The registration yields the current tumour pose (i.e. translation and rotation) as the radiation beams are accurately aligned to the desired target according to the planned position and orientation of treatment [137][112][29]. Registration is a computationally expensive process that is required to be processed quickly in order to ensure an accurate and quick response from the target location system. This requires many DRR images to be rendered from a pre-computed CT volume and the acquisition of X-ray images (typically at 1 min intervals [137] for Cyberknife system). This type of 2D/3D registration is performed between kV/kV image modalities of X-ray and DRR images. Different studies have investigated the performance of the 2D/3D registration of dignostic images, the so called kV/kV problem [38][37][141][137][63][25]. In Chapter 7 of this thesis we present our method of speeding up 2D/3D image registration of kV/kV images that is applicable to this type of radiation therapy systems.

2.4.3 Tomotherapy

Tomotherapy is the latest radiation therapy system which is capable of delivering IMRT by radiating tumours helically with a combination of the use of sophisticated computer controlled radiation beam collimation and the use of on-board CT scanner for treatment site imaging. The special design of tomotherapy makes it capable of providing unprecedented accuracy in beam delivery to the tumour sites and at the same time reduce beam delivering to the healthy tissues [79]. The general shape of the system looks like a CT scanner but internally it is a radiation therapy system operated in MV X-ray energy as shown in Figure 2.17.

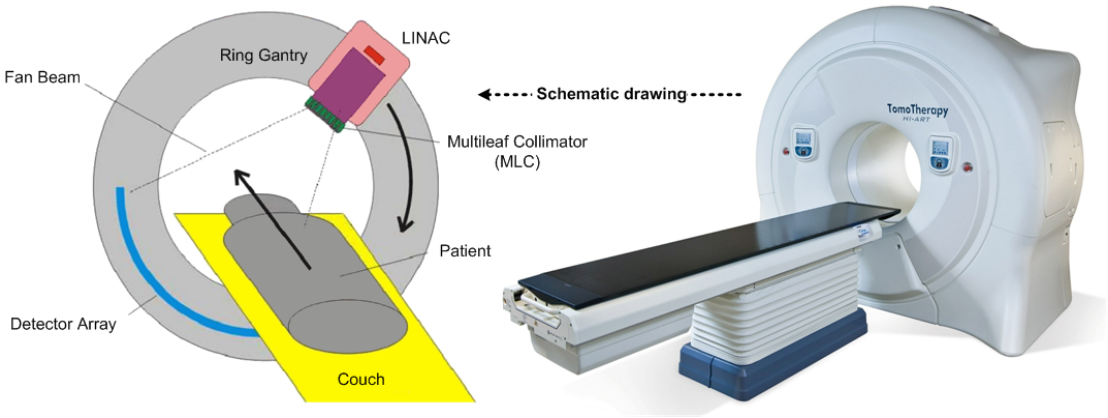


Figure 2.17: A system overview of Tomotherapy® with schematic drawing shows the main components for the helical tomotherapy [189][102].

This system of radiation therapy combines the IMRT delivery with an internal image guided system that uses a megavoltage computed tomography scanning (MVCT). The linear accelerator (LINAC) with 6 MV power is located on a CT is ring gantry that is able to generate a collimated fan beam using binary multileaf collimator (MLC). The MV intensity modulated beam is delivered to the patient from different points through the helical motion of the LINAC around patient's body. Radiation is delivered by continuous rotation of the MV LINAC during the treatment procedure, while the couch is translated at a constant speed inside the gantry. Although the radiation is delivered in different shapes for different slices according to the delivery direction as the tumour usually has irregular shape.

Radiation doses are tumour shaped using the MLC for accurate delivery. However, the system allows an acquisition for low dose MVCT images (i.e. 3.5 MV) that are used in the image registration for the patient positioning process. Each of the MVCT images is collected using array of detectors moving in the same gantry of the LINAC as illustrated earlier in Figure 2.17 [189].

Unlike the other radiation therapy systems, the tomotherapy's patient positioning process is performed without a need for the DRR rendering, as the image registration performed for a pre-collected kilovoltage computed tomography (kVCT) slices and low dose MVCT slices. Patient positioning is performed before each of the treatment fraction by collecting MVCT images (i.e. using the same source of radiation for the treatment with low dose) with different slice spacing (i.e. 2, 4 or 6 mm) to be registered (aligned) with kVCT images that have been used in the treatment planning phase [78]. Therefore, we can conclude that patient positioning with the tomotherapy system is not going to directly benefit from our fast 2D/3D image registration but our work on image registration using a sparse set of kVCT and MVCT images might be useful and worthy of investigation in future work. Different studies investigating the accuracy and speed of the registration in the tomotherapy are discussed by [78][33][34][188].

2.5 Summary

In this chapter we described different medical image modalities that are related to our research, focusing on X-ray images which is one of the main modalities. The method of generating X-ray images and the different energies used affects the quality of images. These are important issues that affect some of the results and artefacts presented throughout the following chapters. Moreover, we described different systems of intensity modulated radiation therapy that uses 2D/3D image registration which is the particular focus of this thesis.

Chapter 3

2D/3D Medical Image Registration.

In this chapter we examine 2D/3D rigid image registration and all its components. In Section 3.1, we define image registration as a rigid transformation and show a schematic overview of 2D/3D image registration. In Section 3.2 we examine three popular similarity measures used in image registration and describe the algorithm we used in our implementation. In Section 3.3 we examine two categories of optimisation methods for image registration and illustrate the method that we implemented in our registration system. In Section 3.4 we examine the DRR rendering process and illustrate methods of rendering the DRR images. Rendering DRRs is a computationally expensive process that forms a focus of much of the experimental work in this thesis. We introduce four different methods of speeding up the rendering of DRR images and illustrate each. In Section 3.5 we discuss different methods of enhancing the performance of the 2D/3D medical image registration.

3.1 Introduction

Image registration is the determination of a geometrical transformation that aligns features in one view of an image or volume with corresponding features in another view of the same or another image or volume [158]. In medical imaging regis-

tration, corresponding images could have the same or different modality, common modalities are Computed Tomography (CT), Magnetic Resonance (MR), Single-Photon Emission Computed Tomography (SPECT) or Positron Emission Tomography (PET). Registration between like modalities, such as MR-MR, is called "intramodal" or "monomodal" registration, whereas registration between different modalities, such as CT-MR, is called "intermodal" or "multimodal" registration [158]. Our research studies intramodal registration between X-ray images and CT images. In image registration a geometrical transformation T is applied to a point x to produce a transformed point $\grave{x}$:

$$\grave{x} = T(x)$$

Successful registration will make point x equal to $\grave{x}$ or approximately equal (in the geometrical meaning), otherwise there will be a registration error if the displacement between x and $\grave{x}$ is not zero. Geometrical transformation is partitioned into rigid transformations which are defined as geometrical transformations that preserve all distances (i.e. translation and/or rotation) and non-rigid transformations (i.e. rigid plus scaling, affine, projective, perspective and/or curved transformations, for more details see [158]). In our research we only studied registration with rigid transformation as our focus in not to investigate the registration field in general, but to establish a registration framework to evaluate our methods of accelerating DRR rendering and 2D/3D image registration. Moreover, the problem that we are targeting (radiation therapy) is mainly a uni-modal registration problem that does not really warrant the extra complexity of a non-rigid framework (Chapter 7 demonstrates that acceptable limits for clinical target registration error (TRE) can be achieved by rigid registration).

3.1.1 Rigid Transformation

A rigid transformation is defined as a geometrical transformation that preserves relative distances (i.e. if P and Q are transformed to $\grave{P}$ and $\grave{Q}$ then the distance from P to Q is the same distance form $\grave{P}$ to $\grave{Q}$) [158]. The components of rigid transformation are translation and rotation, where the translation is a three dimensional vector t with components of (t_x,t_y,t_z) relative to the Cartesian axes (x,y,z)

and rotational components R parameterised in term of three angles of rotation (θ_x,θ_y,θ_z) about the Cartesian axes (x,y,z), often called "Euler angles" [44][90]. Then if T is rigid,

$$\grave{x} = R(x) + t$$

The matrix representation t of the translation (t_x,t_y,t_z) is

$$t(t_x, t_y, t_z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and the matrices representing R (R_x,R_y,R_z) are

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) & 0 \\ 0 & sin(\theta_x) & cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\theta_y) = \begin{bmatrix} cos(\theta_y) & 0 & sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\theta_z) = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

t and R can be combined into a rigid transformation matrix T, written

$$T = t(t_x, t_y, t_z) \times R_x(\theta_x) \times R_y(\theta_y) \times R_z(\theta_z)$$

$$= \begin{bmatrix} c_y c_z & s_x s_y c_z - c_x s_z & c_x s_y c_z + s_x s_z & t_x \\ c_y s_z & s_x s_y s_z + c_x c_z & c_x s_y s_z - s_x c_z & t_y \\ -s_y & s_x c_y & c_x c_y & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where,

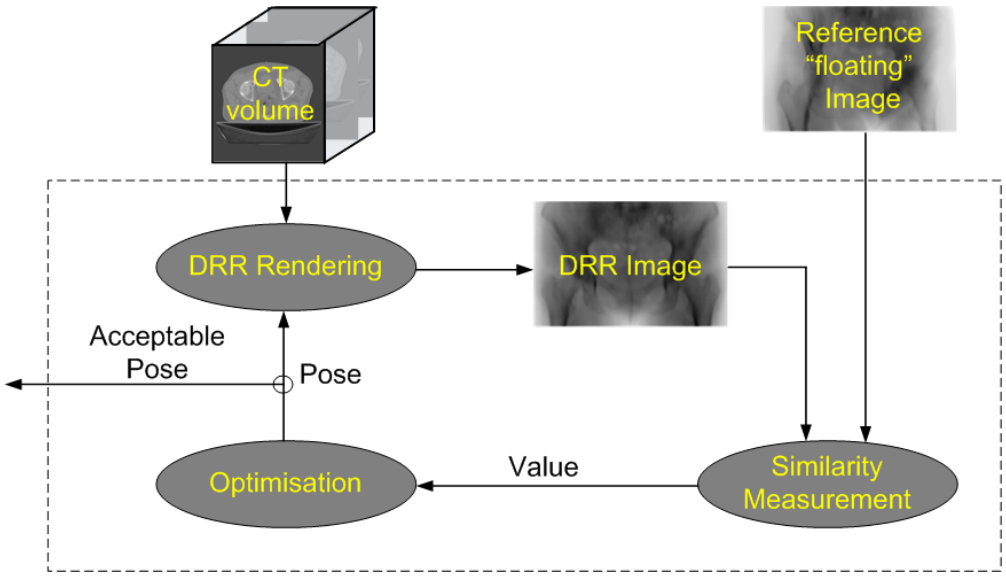
$$s_x = sin(\theta_x), s_y = sin(\theta_y), s_z = sin(\theta_z), c_x = cos(\theta_x), c_y = cos(\theta_y), c_z = cos(\theta_z)$$

Rigid registration comprises searching for the optimum six parameters of T that best aligns the coordinate systems of the corresponding images [46][10].

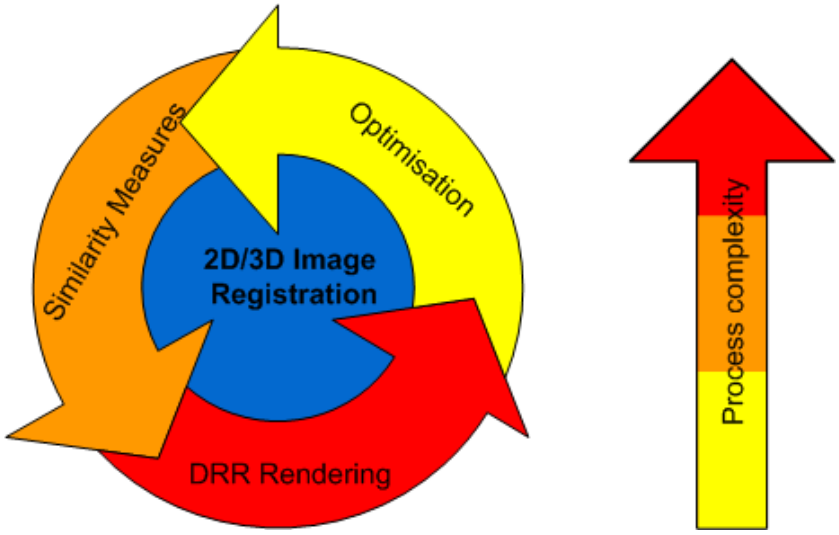
3.1.2 2D/3D Registration Schematic Overview

2D/3D medical image registration, an image guided procedure [187] used to match preoperative images and plans to images captured intraoperatively, is used to align the patient prior to the delivery of radiation therapy treatment [104]. 2D/3D image registration registers portal X-ray images acquired immediately prior (or during) radiation therapy treatment to Digitally Reconstructed Radiograph (DRR) images, rendered from CT volumetric data (more details in Section 3.4). It returns a rigid transformation T (rotation and translation) which provides parameters that are used to position the couch, thereby aligning the patient's anatomy with that used for the planning process [11][39]. An iterative schematic overview of 2D/3D medical image registration is shown in Figure 3.1(a).

The schematic overview of the registration process illustrates the use of a similarity metric to determine the similarity between the reference image and the DRR images (more details about similarity measures presented in Section 3.2). The similarity value is used by the optimisation algorithm to search for T values that generates the most similar DRR image to the reference image (more details about optimisation methods presented in Section 3.3). The registration process is iteratively repeated until optimal values of T are found. The rendering of a number of DRR images is a vital step in each iteration of the registration process. DRR rendering is an extremely computationally expensive process and forms a bottleneck in 2D/3D image registration [84][107]. Figure 3.1(b) illustrates the



(a)



(b)

Figure 3.1: (a)Iterative schematic overview of 2D/3D medical image registration [182], (b) graphical representation for complexity of the 2D/3D image registration.

complexity of the DRR rendering process relative to other 2D/3D image registration components (more details about the DRR rendering methods is presented in section 3.4)

3.2 Similarity Measures

As illustrated in Section 3.1.2 and the schematic overview of 2D/3D registration (Figure 3.1), similarity measurement is a vital process in the iterative workflow. The similarity yields a measurement value that indicates how well the reference X-ray image matches the rendered DRR image. Similarity measurement in 2D images is a fundamental and well researched image processing problem and as such, a number of authors have surveyed the literature in this area [146][123][177]. The similarity measure methods are mainly categorised into two classes; intensity-based and feature-based similarity measures. Feature-based measures requires some pre-processing or user interaction with the images and include the use of landmarks, corner detection and segmentation in order to obtain significant information [182][106]. On the other hand, intensity-based measures that only require pixel intensities use images without the need for pre-processing [182]. As the main goal of this research is to speed up 2D/3D registration, we will use only intensity-based similarity measures as these do not require any pre-processing. The following section introduces the similarity measures we used in our research (i.e. Normalised Cross Correlation (NCC) and Sparse Normalised Correlation(SNC)). We also describe Mutual Information (MI). Although we did not use this technique to measure the similarity between images, it is included to introduce the concept of entropy which was used to find regions of interest in a sparse rendering algorithm we developed (Chapter 7).

3.2.1 Normalised Cross Correlation

Normalised cross correlation is used in many different applications of computer vision when there is a need to assess the similarity between two data sets. It is sometimes simply called a correlation coefficient when the data sets are images as in 2D/3D registration [40][90]. NCC is one of the simplest and most effective

methods of measuring the similarity between images. The main advantage of NCC is that it is invariant to linear changes in image intensity (i.e. if the pixels intensities in one of both images are scaled by a constant value, then the NCC value will not be changed). NCC is defined as:

$$NCC(I_{ref}, I_j) = \frac{\sum_{i=1}^{N}(A_i - \overline{A}) \cdot (B_i - \overline{B})}{\sqrt{(\sum_{i=1}^{N}(A_i - \overline{A})^2) \cdot (\sum_{i=1}^{N}(B_i - \overline{B})^2)}} \tag{3.1}$$

Where A represents the reference image I_{ref} and B a floating DRR image I_j, $\overline{A}$ and $\overline{B}$ are the mean intensity values and N is the total number of image pixels.

But using Equation 3.1 we need to read both of the images two times in order to calculate the mean intensity and to complete the summation. Therefore to reduce the calculation time our application used an optimised version of NCC (i.e. NCC′) presented by Wolfganag [182] that allows us to read both of the images for once only, by expanding the equation into the following form:

$$NCC'(I_{ref}, I_j) = \frac{(\sum_{i=1}^{N} A_i B_i) - N\overline{A}\,\overline{B}}{\sqrt{(\sum_{i=1}^{N} A_i^2) - N\overline{A}^2} \cdot \sqrt{(\sum_{i=1}^{N} B_i^2) - N\overline{B}^2}} \tag{3.2}$$

In this case both of the images have to be read only once. Additionally, in 2D/3D registration cases where only one image changes in each iteration (i.e. the floating DRR image we described earlier in Section 3.1.2), there are considerable advantages in using this method.

3.2.2 Sparse Normalised Correlation

One of our approaches speeds up the 2D/3D registration loop by rendering only parts of the images; we call these images sparsely rendered DRRs. We adapt the method of NCC to provide an efficient similarity metric that is applicable to sparse images, calling it Sparse Normalised Correlation (SNC). SNC enables us to measure the similarity for a sparse set of regions (i.e. the whole of the image is not covered) and thereby register these, as shown in Figure 3.2. Formally this can be represented as Equation 3.3.

$$SNC(I_{ref}, I_j) = \sum_{i \in S} NCC'(I_{ref}, I_j, i) \tag{3.3}$$

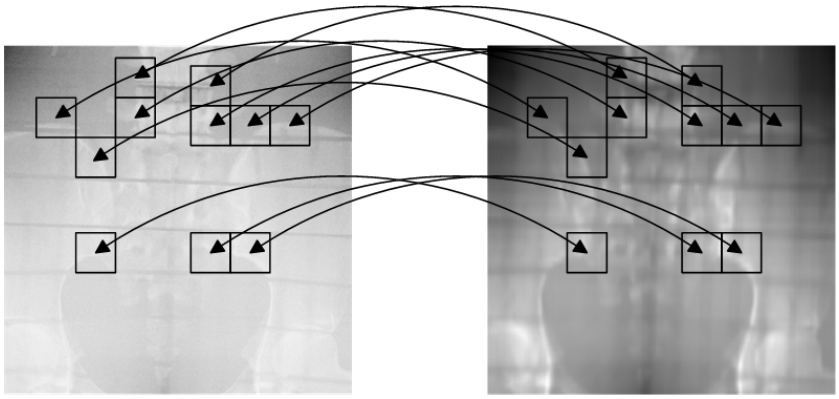


Figure 3.2: Illustrates the SNC method of similarity measure between an X-ray image and a DRR image.

Where I_{ref} represents the reference image, I_j a floating DRR image , S represents the sparse regions in the image and i is regions' pixels counter.

A similar approach is presented by LaRose to provide a similarity metric that is robust to non-linear changes of intensity, called Local Normalised Correlation (LNC). According to LaRose [90] the idea is to find the mean NCC over a set of regions R covering the image, as shown in Figure 3.3. Formally this can be represented as Equation 3.4.

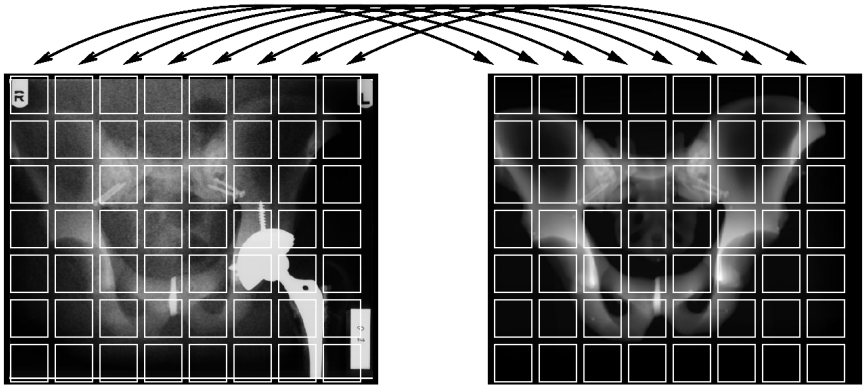


Figure 3.3: Illustrates the LNC method of measuring the similarity between an X-ray reference image and a DRR image [90].

$$LNC(I_{ref}, I_j) = \frac{1}{|R|} \sum_{P \in R} NCC(I_{ref}, I_j, P(p)) \tag{3.4}$$

Where R is a set of regions covering the image, each comprising $P(p)$ pixels, the summation of LNC is divided by $|R|$ to obtain the mean value of LNC over the image. LNC is useful in cases where one of the images exhibits a non-linearity. Such a case might arise when the patient is fitted with a metal prosthesis, as shown in Figure 3.3 (left-hand side).

In our implementation of the similarity measure, we used the modified NCC (NCC′ Equation 3.2) inside the equation used to compute SNC (Equation 3.3). This approach is more efficient over the method of LNC which proposed by LaRose [90] for reasons we already discussed. Moreover, we compute the SNC for part of the image instead of whole the image by selecting ROI automatically (as shown in Figure 3.2 and illustrated in Section 7.3.2). This gives us further efficiency gains over the LNC method in the 2D/3D image registration because we only need to calculate the similarity measure for selected areas. More details about the implementation method for the 2D/3D image registration and the SNC of measuring the similarity between ROI for the reference image and the DRR images is presented in Chapter 7.

3.2.3 Mutual Information

Mutual Information (MI) measures the amount of shared information by expressing the statistical correlation between images using the entropy measure. Shanon's entropy is used to asses the amount of information in an image according to Equation 3.5 [56][182] (more theoretical information about the entropy is presented in Chapter 7):

$$E(I) = -\sum_{i=0}^{255}(p(i) \times log\ p(i)) \tag{3.5}$$

Where $p(i)$ is the probability of the i th intensity value of the histogram ($i = 0 \rightarrow 255$) for the local regions of our gray scale images I, $p(i) = Histogram\,[i] \div regionSize$. To asses the amount of information shared between two images, the joint entropy is used as presented in Equation 3.6 [98].

$$E(I_{ref}, I_f) = -\sum_{i,j=0}^{255} p_{A,B}(i,j) \times log\ p_{A,B}(i,j) \tag{3.6}$$

Where A represents the reference image I_{ref}, B the floating DRR image I_f and $p_{A,B}(i,j)$ is the joint probability distribution of A and B. Intensity probability distribution can be visualised as a two dimensional joint histogram as in Figure 3.4. MI combines the individual and joint entropy of images as in Equation 3.7.

$$MI(I_{ref}, I_f) = E(A) + E(B) - E(A,B) \tag{3.7}$$

Where A represents the reference image I_{ref}, B the floating DRR image I_f, $E(A)$ and $E(B)$ denote the individual entropy and E(A,B) denotes the joint entropy. MI is one of the most frequently used methods in the intensity based registration, especially to measure the similarity between different image modalities. The literature on MI demonstrates that robust and accurate results can be obtained as MI assumes there are no functional dependences between the images, only a statistical dependence between their intensities . However, the main drawback that has limited the use of MI in some applications is that it does not consider spacial information [126][100][98]. The MI measure is attractive since it can be easily calculated from the image histogram, but the computational cost of the technique is higher compared with NCC and since our images are uni-modal there is no advantage to be gained by using this approach. However, our sparse rendering algorithm uses an entropy measure to select ROI required (as we illustrate in Chapter 7).

3.3 Optimisation

As we briefly illustrated in the schematic overview of the 2D/3D image registration (Section 3.1.2), we use an optimisation process to find the parameters for the rigid transformation T in order to correctly register the reference image with the DRR image. Optimisation algorithms work iteratively to search the parameter space T, and terminate once either the accuracy criteria has been satisfied or a local/global minima/maxima has been found [158][182]. In general optimisation algorithms are search algorithms that cannot guarantee 100% perfect results in terms of accuaracy, computaional time and speed, but they do have the ability to find an optimal solution [160]. Spork [160] illustrates this by an example which

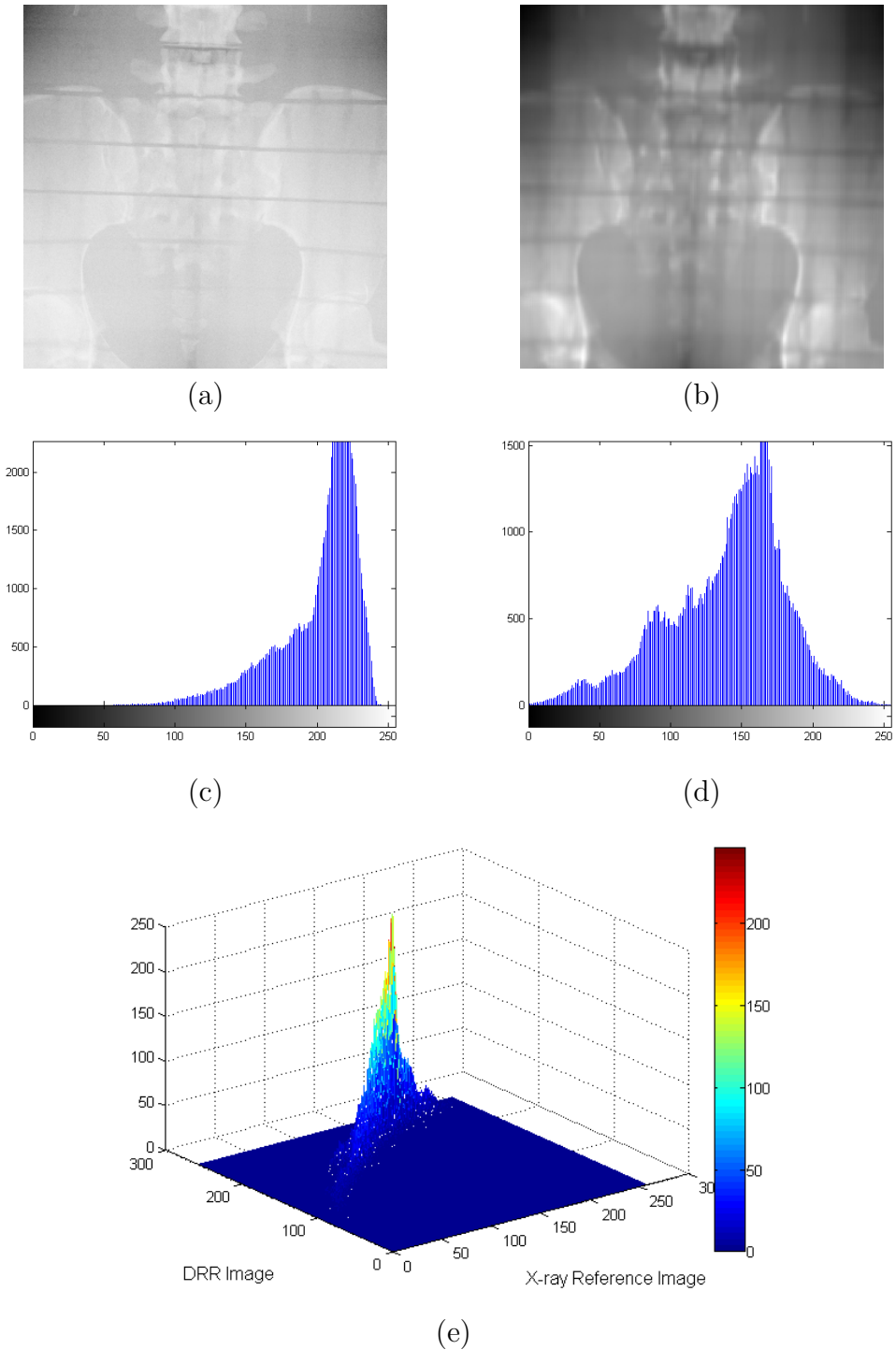


Figure 3.4: Illustrates the joint histogram used in MI for 2D/3D image registration. Where (a) shows an X-ray reference image, (b) shows a DRR image, (c) shows a histogram for the reference image, (d) shows a histogram for the DRR image, and (e) shows a joint histogram built from the two histograms for the reference and DRR images.

attempts to register two non identical images. In this case, none of the similarity metrics will indicate a perfect match but the optimisation framework will try to find the best solution. Optimisation algorithms are divided into two main categories, either gradient based algorithms (e.g. gradient descent [53]) or non-gradient base algorithms (e.g. best neighbour search [171]). A brief introduction for the best neighbour search algorithm and the gradient descent algorithm is presented in the next two sections.

3.3.1 Best Neighbour Search

The best neighbour search algorithm (hill-climbing) refers to the family of pattern search algorithms. In each iteration of the algorithm the position of the floating image is altered by a specific step size in the parameter space according to the result of the cost function [182]. When used in 2D/3D image registration the similarity measure algorithm (cost function) is used to evaluate the similarity between the reference image and the DRR image at a specific location in a parameter space comprising six degrees of freedom (DoF) [173]. After comparing 12 DRR images (i.e. both directions for each DoF 2×6) the one with the highest similarity value will be chosen and set as the base step for the next iteration. If non of the compared DRR images achieves a better value than the current one, either the algorithm will terminate or it will downscale the step size and re-execute the iteration in order to find the optimal position of the volume used to generate the DRR images [182]. The optimal position should be the global maxima for the optimisation algorithm. However, a drawback of this approach is that the search algorithm can get stuck at a local maxima, instead of continuing to find the global maxima [103]. Finally, although best neighbour search is one of the simplest optimisation algorithms, recent results presented by [141] show that it is a very suitable algorithm for optimisation in the 2D/3D image registration.

3.3.2 Gradient Descent

Gradient descent is a gradient based approach used in the optimisation process in order to find the global optima by successively stepping in the direction of the function's gradient as illustrated in Figure 3.5.

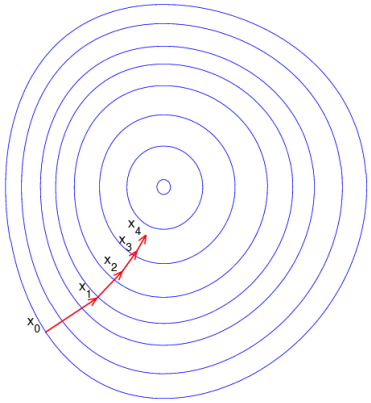


Figure 3.5: Example illustrates the gradient descent method [5]

The success of the approach depends mainly on selecting the correct size of the step at each iteration according to the gradient function:

$$x_{t+1} = \alpha \frac{df(x_t)}{dx_t}$$

Where α is the step size of the function. α should be chosen carefully; if the step size is too big this could take the algorithm far from the global optima or if the step size is too small the algorithm could get stuck in a local optima, never reaching the global optima [182].

3.3.3 Our Implementation

In a comparable study published by Khamene et al. [84] examining different optimisation techniques (e.g. the best neighbour search, gradient descent, powell-brent) for 2D/3D registration, the mean and standard deviation of TRE showed no significant difference irrespective of the different optimisation technique used. They conclude that the type of optimisation technique does not affect the results of the 2D/3D image registration. Therefore, from our point of view the most important issue in the choice of optimisation technique is the number of DRR images required in order to reach the global optima.

We implement a solution for the optimisation problem by posing the six degree of freedom as 3 independent optimisation problems in two degrees of freedom.

These are solved by fitting a 2^{nd} order polynomial function to the data and differentiating to find the maxima. This reduces the required number of DRR images that need to be rendered, compared to a hill-climbing algorithm and thus reduces the time required for registration [39]. For each of the 3 optimisations the parameters (x, y, z, yaw, pitch, roll) were paired (another research [123] suggests to combine multiple DOFs). As much as possible we choose uncorrelated pairs i.e. {x, pitch},{y, yaw},{z, roll}. For each pair we render five DRRs and compute the similarity metric for each. According to our optimisation strategy 15 DRRs (3×5) need to be rendered. This is slightly more than needed at each iteration of an equivalent conventional best neighbour search. However, one must remember that the approach is not iterative, once the polynomial surface has been fitted the maxima can be found in one step. More details about our implemented method of optimisation for the 2D/3D image registration is illustrated in Chapter 4, Section 4.4.

3.4 Digitally Reconstructed Radiographs (DRRs)

A Digitally Reconstructed Radiograph (DRR) is a two dimensional simulated X-ray image, rendered from medical tomography data sets, such as Computed Tomography (CT) [9]. Rendering DRR images is important for many medical applications, such as, 2D/3D medical image registration [119] and brachytherapy [109]. In radiation therapy treatment systems, floating DRR images are a vital part of the patient positioning process and may be aligned manually or automatically. DRRs are rendered from the medical tomography data by summing the attenuation due to each voxel along known ray paths through the data volume. However, this conventional ray tracing approach to DRR rendering is an extremely computationally expensive process and forms a bottleneck in medical applications, like 2D/3D image registration [84][107]. Normally, conventional DRR rendering requires $p \times q$ rays to be cast to generate a DRR from a data volume; where p and q are determined by the image resolution. In patient positioning this is usually chosen to match that of the solid-state flat panel X-ray detector as illustrated in Figure 3.6.

The complexity of DRR rendering results from the massive number of calcu-

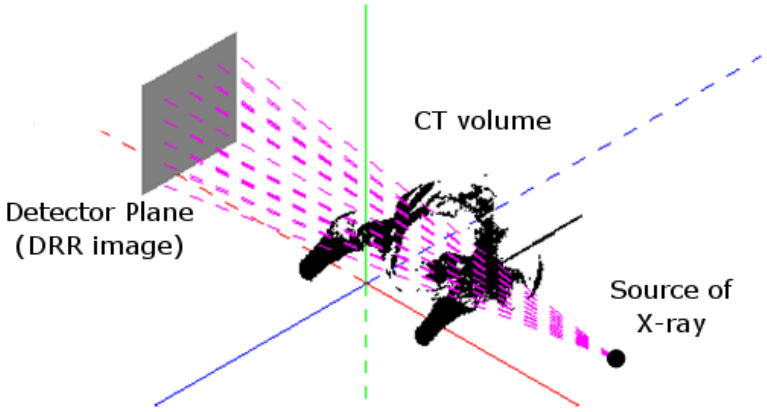


Figure 3.6: Illustrating the geometry of DRR rendering.

lations needed and the large number of ray casting operations. Compared with more general surface rendering techniques, rendering DDR images is considerably more computationally demanding as we need to compute the attenuation of a monoenrgetic beam due to different anatomic material (e.g. bone, muscle tissue, etc.) within each voxel, using Beer's Law [83].

$$I = I_0 * exp^{\Sigma - \mu_i x_i}$$

Where I_0 is the initial X-ray intensity, μ is the linear attenuation coefficient for the voxel (material) through which the ray is cast, x is the length of the X-ray path and subscript i denotes the voxel index along the path of the ray, as illustrated in Figure 3.6. Voxel values in CT volumes are represented by a CT number quantified in Hounsfield Units (HU). The attenuation coefficient of the material comprising each voxel can be recovered by [148]:

$$CTnumber = 1000 * [(\mu_i - \mu_w)/\mu_w]$$

where μ_i is the attenuation value of a particular volume element of tissue (voxel) and μ_w is the linear attenuation coefficient of water for the average energy in the CT beam.

Various methods have been proposed to speed up the rendering of DRR images. Through the following sub-sections we will cover some of the most recent and important methods of speeding up DRR rendering. We explored four methods for

DRR rendering. The first is a simple ray casting method. This has been chosen by other researchers as a reference method in studies comparing speed, accuracy, and quality [95][90][141]. The second method we chose uses a technique known as attenuation fields (AF) [141]. Which represents a recent DRR rendering method which claims to offer improvement over ray casing methods. The third method is the shear warp method, which is another fast and accurate method of rendering volumes [110]. The fourth is a hardware based method which demonstrates some advantages of using graphics hardware in the rendering process in order to improve the speed. Finally, we then describe our method for rendering of DRR images.

3.4.1 Ray Casting

Ray casting is one of the oldest and most straight forward methods for rendering 3D data volumes. It simulates real light rays that strike objects by finding their intersections. In general purpose volume rendering, ray casting is used to find surface intersection points between the rays and the objects. The surface is identified by defining a threshold value. The volume is visualised by casting the volume front to back and rendering the first voxel having value above the threshold [184][160]. However, in the case of DRR rendering each of the rays will intersect all voxels inside the 3D data volume which are located in the direction of the ray. Figure 3.7 illustrates both of the methods. The general concept of any ray casting algorithm for volume rendering is to have a ray with a specific direction that intersects a 3D data volume. A specific ray r can be mathematically described as:

$$\vec{r}(t) = \vec{o} + t\vec{d}$$

Where o is the origin of ray source, d is the direction of the ray and t determining the ray length (i.e. $0 \leq t < \infty$) [61]. The directions of the rays differ according to the desired projection style (orthogonal, perspective). In orthogonal projection, the rays are perpendicular to the image plane and parallel to each other, while in perspective projection, the rays have the same source (origin of ray source) [107]. Origin of the ray and this geometry matches that of the Varian LINAC system used to acquire the reference X-ray images.

Data sampling was one of the earliest proposals aimed at reducing the compu-

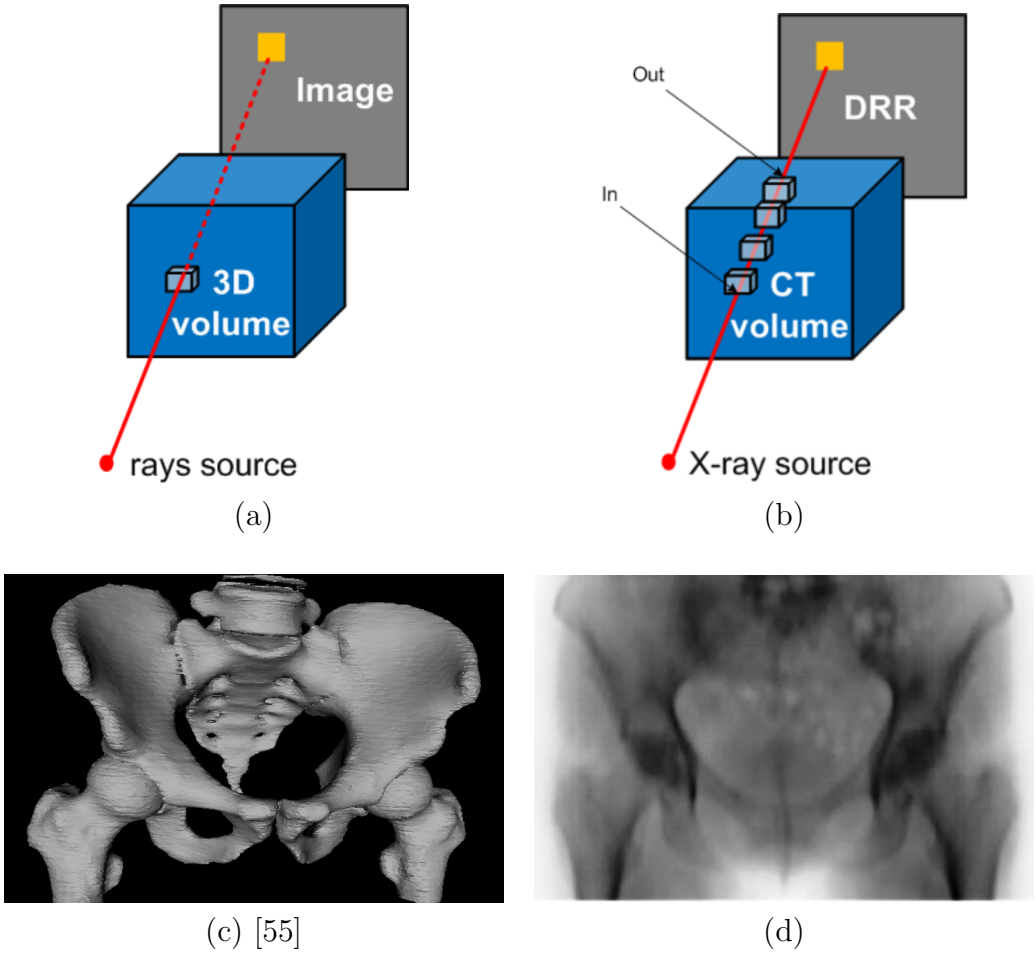


Figure 3.7: Illustration of the difference between visualising a 3D data volume and DRR rendering from a 3D data volume using ray casting method. Where (a) shows a ray casting method for 3D volume surface rendering, (b) shows a ray casting method for DRR rendering from a 3D data volume, (c) shows an example of the 3D volume surface rendering method and (d) shows an example of the DRR rendering method.

tational cost of DRR rendering . This method increases the performance of the ray casting method but at the same time it has some drawbacks that motivates researchers to develop different methods of ray casting. One of the main drawbacks is related to the accuracy of the DRR image resulting from the rendering process as some of the small structures in the volume can easily be missed. Also the time needed to render a volume is fixed by the number of samples taken and many of these may lie in the space surrounding the object. This space is likely to be air, which has comparatively little effect in terms of X-ray attenuation. To avoid sampling in empty spaces authors proposed a way to collect the empty spaces (voxels) in a large block of data using the Octree/kdtree data structure [94][159][39][183]. More details about the method of using the Octree in the rendering of DRR images is illustrated in Chapter 4. On the other hand, the ray casting algorithm is simple to design and implement. There is also an advantage of using ray casting in DRR rendering as each pixel of the DRR image is calculated independently (i.e. we accumulate all the intersection points located in the direction of the ray) and therefore ray casting is perfect for parallelisation [107]. Many research methods have been developed to parallelise ray casting algorithms using the hardware (Graphical Processing Unit) [51][76][98][111]. In our research we explore parallelising the ray casting algorithm using both CPU and GPU. We illustrate these approaches in Chapter 5 and Chapter 6 respectively.

3.4.2 Attenuation Fields

An attenuation field (AF) is a special data structure implemented by Russakoff et al. [141] used in the rendering of a DRR instead of the conventional ray casting methods. According to the original proposal of light fields by Levoy and Hanrahan [93] and similar work in concept (transgraphs) introduced by LaRose [90], AFs provide a way of parameterising the set of rays that emanate from a static scene to perform 3D rendering. Rays in the rendering space are parameterised as $R \equiv Pi(u, v, s, t)$ where plane (u,v) is the focal plane and (s,t) is the virtual image plane (camera plane) as illustrated in Figure 3.8. An important feature of the parameterisation is that it can only cope with small relative movements of the camera (X-ray source) and CT volume.

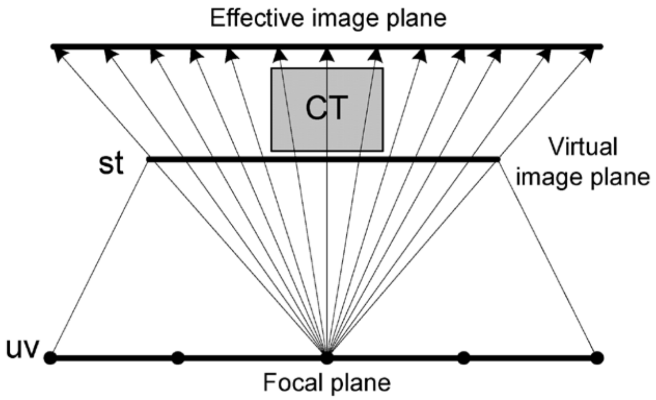


Figure 3.8: DRR rendering using AFs [141].

Images of objects inside the light slab (which is a convex quadrilateral object, formed by two planes (u,v) and (s,t)), are created by calculating a huge number of rays (theoretically an infinite number of rays) inside this light slab.

In Russkoff's application of AF DRR rendering [141], the static 3D scene used to illustrate the approach in Levoy and Hanrahan's work [93] is replaced by a 3D CT volume and DRR images are rendered by calculating the integral linear attenuation coefficient affecting rays passing along paths from the X-ray source to the flat panel detector (destination). To accommodate the differences between the original structure and the altered one, a virtual image plane is introduced to model this *projective* geometry. Comparisons are drawn between this imaging geometry and that originally proposed by Levoy and Hanrahan in Figure 3.8. A large number of rays inside the light slab were rendered (i.e. not *all* the rays) and then the missing values were estimated by interpolation.

To build the light slab, a set of DRR images for a limited range of gantry/couch positions (view sphere) must be rendered offline in an earlier step in order to create the AFs. Once it is built, the AF look-up-table (LUT) can be used to quickly generate DRRs from novel view-points within the view sphere using an interpolation of the 4D ray space we illustrate earlier. There is an obvious memory saving in using AFs, over precomputing a complete set of DRRs covering every possible gantry/couch position. However, the size of the AF LUT is in the order of approximately $\approx$ 2GB (i.e. (u,v) resolution of 64×64 pixels and (s,t) resolution of 512×512 pixels). To address this, Russakoff also compressed the AFs using vector

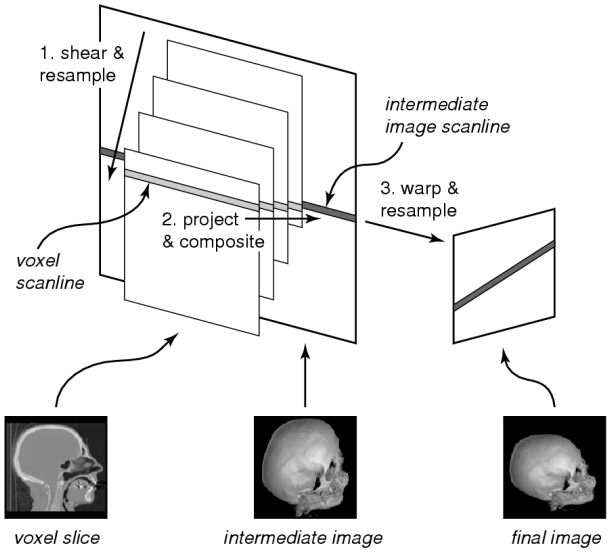
quantisation, claiming a compression ratio equal to $\frac{kb}{\log_2 N}$ where k is the number of elements per vector, b is the number of bits per element, and N is the number of codewords.

3.4.3 Shear Warp Factorisation

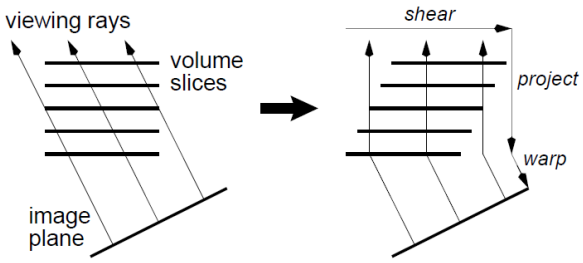
Shear Warp Factorisation is an object space based volume rendering algorithm proposed by Lacroute et al. [125] to achieve interactive rendering rates without significantly affecting the image quality. The shear warp factorisation algorithm consists in general of three steps (see Figure 3.9(a)). The first step, shears and resamples the volume slices to form a set of intermediate 2D image slices. The second step, projects the resampled voxel scanlines onto the intermediate image slices and the third step, warps the intermediate image into the final result. To transform the object into the shear object space (step one), there are two cases according to the direction of the rays. If the rendering is orthogonal then the volume is transformed to sheared object space by translating each slice. If the rendering is perspective then the volume transformed to sheared object space by translating and scaling each slice as illustrated in Figure 3.9(b) and 3.9(c) respectively. The main advantage of shear warp factorisation is that scanlines of the volume data and scanlines of the intermediate image are always aligned. Which allows efficient and synchronised access to the data in the volume to render the final image [125]. A shear warp factorisation was used in DRR rendering by Wesse et al. [181] in their research on 2D/3D image registration. More details about their work are illustrated in the related work section (Section 3.5).

3.4.4 Hardware Texture Mapping

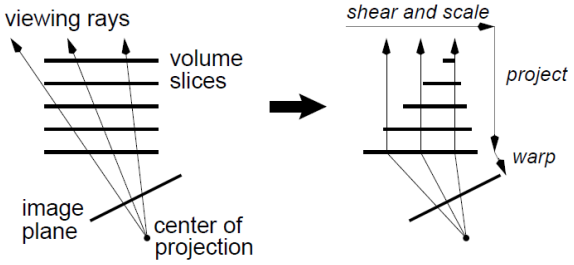
Hardware based volume rendering is a popular topic for research in the graphics area. The implementation of hardware based solutions for volume rendering depends mainly on the algorithm used for volume rendering. Hardware is no more than a special tool used to achieve efficient solutions and enhanced results for the implemented rendering algorithms. Parallelising the implementation of the rendering algorithm on the hardware device is the main objective for researchers wishing to take advantage of the large number of processing cores located on



(a)



(b)



(c)

Figure 3.9: Illustration for the shear warp factorisation algorithm. Where (a) shows the three steps of a shear warp algorithm, (b) shows a volume transformed to shear object space for an orthogonal projection by translating the slices, and (c) shows a volume transformed to shear object space for a perspective projection by translating and scaling the slices [125].

modern graphics cards. However, to be able to program hardware graphics devices (graphics card) there is a need to use a special programming language or an application programming interface (API). Nowadays there are many choices of APIs and programming languages (e.g. OpenGL, Direct3D, CUDA, Cg, etc.) used with different types of hardware (e.g. NVidia, ATI). Graphics hardware comprises a special graphics processing engine (usually employing multiple processing cores) and local/global memory configured to store textures and vertices. 3D scene data is transferred to the graphics card by DMA transfer and rendered to the screen. To investigate DRR rendering using the hardware we have to load the CT volume into the hardware memory to be able to easily and quickly access the CT values. A solution to this problem used to be performed by using a 2D texture mapping to project the values of the CT slices to the hardware memeory [107]. These days 3D texture mapping is used to project the whole CT volume at once. More details about the hardware APIs, programming languages, previous work on DRR rendering and our method of implementing the DRR rendering are illustrated in details in Chapter 6.

3.4.5 Our Implementation of Rendering DRR Images

Our early implementations for rendering the DRR images used two main algorithms. The first algorithm is a ray-box intersection algorithm which we implemented to find the rays that intersect the CT volume and to find the In/Out intersection points (see Figure 3.7). The second algorithm is a point-based sampling algorithm which we implemented to provide estimate for the internal intersection points between the rays and the CT volume. In detail, we implemented an efficient and robust ray-box intersection algorithm that was initially proposed by Williams et al. [8], improving the original algorithm developed by Smits [156]. We used Williams' algorithm to check each ray used in the DRR rendering is intersecting the CT volume and once the In and Out intersection points are known the second point sampling algorithm. We implemented a customised version of the point-based rendering algorithm which was originally developed by Shen et al. [9] and used a conventional sampling instead of the hybrid sampling method (for more details see [9]). Our implementation of the point-based rendering casts rays using

the sampling method once the In/Out intersection points are known and a variable sampling distance is chosen according to the required DRR resolution.

The algorithm 4 described in Appendix B illustrates the fast ray-box intersection, originally developed by [8] that we implemented for the DRR rendering. However, the following algorithm (Algorithm 1) illustrates the DRR algorithm that we implemented in the 2D/3D image registration. Two types of DRR images can be rendered using the Algorithm 1; Full Resolution DRR images (FR-DRR) and Reduced Resolution DRR images (RR-DRR). To render FR-DRR we aim to recovers all the intersection points between the rays and the CT volume by using all the $n \times m$ rays located in rendering field (where n and m are the parameters for the size of the detector plane), with a sampling distance equal to the voxel size. Rendering RR-DRR images is speeded up using two methods: first by reducing the number of rays in the rendering field and second by reducing the number of intersection points inside the CT volume. Figure 3.10 provides a simpler illustration for the different DRR types. More details about the different DRR types and its application in the 2D/3D image registration is presented in Chapter 7.

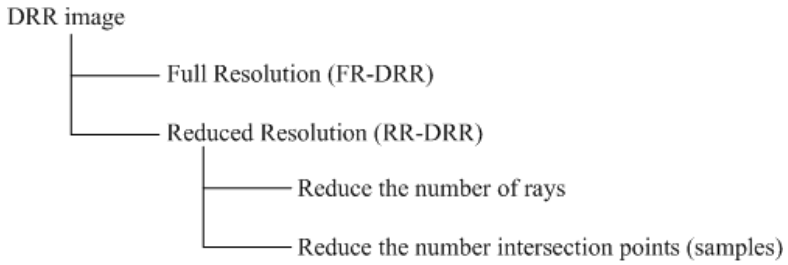


Figure 3.10: Illustration for the different DRR types.

Our implementation of DRR rendering uses a polar coordinate system which models spherical movement around the CT volume and is well matched to the C-arm LINAC geometry. The movement of the detector plane is connected to the movement of the ray source with 180° between the centre of the detector plane and the ray source as illustrated in Figure 3.11.

Algorithm 1 DRR rendering Algorithm: FR-DRR, RR-DRR

```
integer i, j, count
float voxelSize
clock_t start, finish
start← clock()
// Calculate the direction of rays
for all i,count such that 0 ≤ i ≤ img_DimX do
    for all j such that 0 ≤ j ≤ img_DimY do
        temp_point = detectorPlane[count]
        temp_direc = temp_point-x_source
        temp_ray.setRay(x_source,temp_direc)
        x_rays.push_back(temp_ray)
        count = count+1
    end for
end for
//Define the CT volume dimensions
Vector3 min_CT_coord(0,0,0)
Vector3 max_CT_coord(img_DimX,img_DimY,img_DimZ)
Box CT_box(min_CT_coord,max_CT_coord)
// Find the In/Out intersection time for the CT volume
temp_ray=x_rays[0]
t0=0
while !(CT_box.intersect(temp_ray,t0,t0)) do
    t0 = t0 + voxelSize
end while
startIntersectionTime = t0
while (CT_box.intersect(temp_ray,t0,t0)) do
    t0=t0 + voxelSize
end while
endIntersectionTime = t0
// FR-DRR rendering
if full_resolution equals true then
    for all i such that 0 ≤ i ≤ img_DimX do
        for all j such that 0 ≤ j ≤ img_DimZ do
            temp_ray = x_rays[count]
            absorpSum = 0
            t0 = startIntersectionTime
            for all t0 < endIntersectionTime do
                t1 = t0, t0 = t0 + voxelSize
                if CT_box.intersect(temp_ray,t0,t1) then
                    pointPosition=temp_ray.getPointPosition(t1)
                    offset = CTimg.offset(pointPosition.x(),pointPosition.y(),pointPosition.z())
                    absorption = CTimg[offset]
                    absorpSum =absorpSum + absorption
                end if
            end for
            DRR(i,j)=absorpSum
            count = count + 1
        end for
    end for
end if
//RR-DRR images by reducing the number of rays
if full_resolution equals false then
    for all i such that 0 ≤ i ≤ img_DimX do
        for all j such that 0 ≤ j ≤ img_DimZ do
            temp_ray = x_rays[count]
            absorpSum = 0
            t0=startIntersectionTime
            for all t0 < endIntersectionTime do
                t1 = t0, t0 = t0 + voxelSize
                if CT_box.intersect(temp_ray,t0,t1) then
                    pointPosition=temp_ray.getPointPosition(t1)
                    offset = CTimg.offset(pointPosition.x(),pointPosition.y(),pointPosition.z())
                    absorption = CTimg[offset]
                    absorpSum =absorpSum + absorption
                end if
            end for
            DRR(i,j)(i+1,j),(i,j+1),(i+1,j+1)=absorpSum
            count = i × img_DimZ + j
            j = j + 2
        end for
        i = i + 2
    end for
end if
```

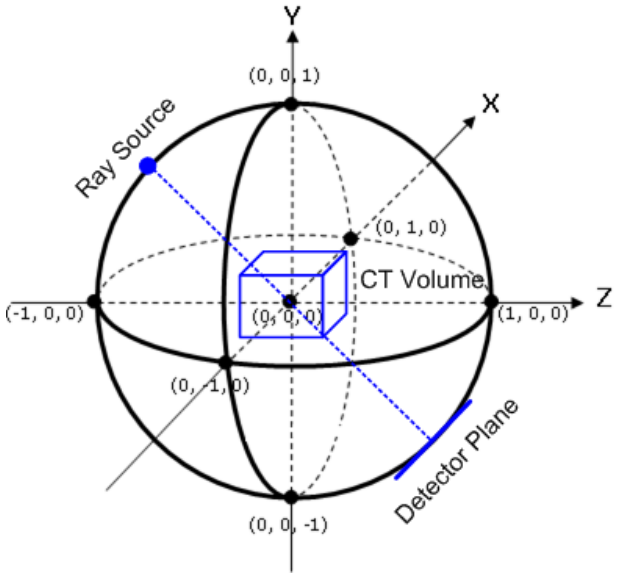


Figure 3.11: The spherical coordinate system used to render DRR images.

3.5 2D/3D Image Registration: Related Work

Various methods have been proposed by many researchers to enhance the performance of 2D/3D medical image registration. In this section we will illustrate the main methods of enhancing the performance of 2D/3D image registration, discussing the results of these methods and demonstrating the internal processes of 2D/3D image registration and its importance.

Russakoff et al. [141] speed up the 2D/3D registration process mainly by speeding up the rendering of DRR images using the attenuation field (AF) method which we introduced in Section 3.4.2. They also crop the reference image to identify a specific region of interest (ROI). This process implies rendering DRRs only for the ROI (200×200), which also contributes to the speeding up of DRR rendering process. Also, they measured the similarity between the images using an intensity similarity measure by applying Mutual Information (MI) measurement, and they performed the optimisation process by using a simple best neighbour search strategy. As a result of the registration process they report similar registration accuracy to that described in previous research using DRRs formed by ray casting (the difference in overall mean target registration error (TRE) is about 0.1 mm).

Russakoff performed a quantitative comparison of the quality of AF and ray cast DRR images by computing the peak signal-to-noise ratio (PSNR). Results show that PSNR values are greater than 43 dB and from this we can conclude that both types of DRR images are similar. They rendered an AF DRRs with a resolution of 256×256 pixels in about 50 ms on a PC workstation using a 2.2 GHz Intel Xeon processor, and rendered AF-DRRs in a ROI (200×200) pixels in about 30 ms. They perform the whole registration process (which requires about 100-150 iterations of optimisation process) in about 100 seconds on the same machine.

Russakoff's experimental framework was developed using a CyberKnife Stereotactic Radiosurgery system which uses two external X-ray sources to acquire portal images in the kilo electron-volt (kV) range. Consequently, good quality high contrast images will be generated from this system. Mega electron-volt (MV) low contrast portal images are routinely used for patient setup on Linear Accelerator (LINAC) systems (although LINAC simulators usually provide a kV X-ray source). Consequently, it is doubtful that such results could be reproduced using a MV LINAC system. In our experiments we found MV images often contained insufficient detail and the similarity measure did not return a value above a predefined threshold. Moreover, in Russakoff's work, cropping the ROI is performed manually. We believe, in some cases this will lead to degraded performance as the ROI may change as the patient moves and so the registration process may be compromised. Their registration method depends on fedicual markers to perform the registration. In another publication, Russakoff et al., present AF-DRR rendering as an alternative to ray casting, but the following statement suggests a hybrid approach is used "projection values needed for a DRR but not found in the progressive attenuation field (PAF) are computed in demand using a fast ray casting engine..." [137]. From this we conclude that the AF-DRR method is not always able to recover the required DRR images (perhaps because the required view is outside the view sphere).

Weese et al. [181] tried to enhance the speed of 2D/3D image registration by speeding up the rendering of DRRs using shear warp factorization, originally developed by Levoy [125]. They used a shear-warping method of volume rendering to generate the intermediate image by adding the values of a stack of CT slices,

which are parallel to the intermediate image plane and obtained the final DRR image from the intermediate image by applying bi-linear interpolation to it. As a result of this method, they can perform the registration process within 3.3 seconds using a Sun Ultrasparc 300 MHz, for CT volume of size 512×512 pixels. Although, they showed that their applied rendering method does not affect the registration accuracy, target registration errors, compared with ground-truth registration of about 0.5° in rotation, 0.5 mm in translation parallel to the projection plane, and 5-6 mm for perpendicular translation on the projection plane were reported.

Gross [168] enhanced the performance of 2D/3D image registration by developing an algorithm that performs registration in an efficient way without degrading the accuracy or speed. The author focused on applying the registration without any user interaction even without any landmarks on the patient's body. Gross used a GPU based ray casting algorithm to create the DRRs in a quick way by performing the rendering process in two passes: in the first pass, back face culling algorithm is used to gives the texture coordinates of the front intersection for each image pixel. In the second pass, front face culling provides the coordinates of the back intersection. Using the texture coordinates of the intersection points and the sample rate, it is possible to determine the points at which to sample the volume texture. Gross used bone and firm structure to give references for the images which are used as measures the similarity between X-ray and DRR images in a registration process that uses an intensity-based model instead of feature-based one. Gross proposed an adjoint DRR algorithm with algorithmic differentiation to compute exact derivatives for gradient based optimisation. As a result, DRR with 256×256 pixels can be rendered in 0.051 seconds from $512 \times 512 \times 318$ voxels CT volume using the GPU raycaster on a machine with an NVIDIA GeForce Go 7400 TurboCache graphics card with 256 MBytes of video memory.

"Automatic registration of portal images and volumetric CT for patient positioning in radiation therapy" is a significant paper in our area, written by Khamene et al. [84] in 2006. It is significant for us because the authors dealt with all 2D/3D registration components and in realistic way, with operations relevant to a LINAC system (coordinates, beam geometry, transformation, etc.). Logically, they di-

vided the registration process in to three main phases, *calibration*, to estimate beam geometry, *planning*, to define the planned target volume (PTV) using CT data, and *target positioning*, to take and rectify the portal images, in order to "discard the gantry sag" [84], as illustrated in Figure 3.12.

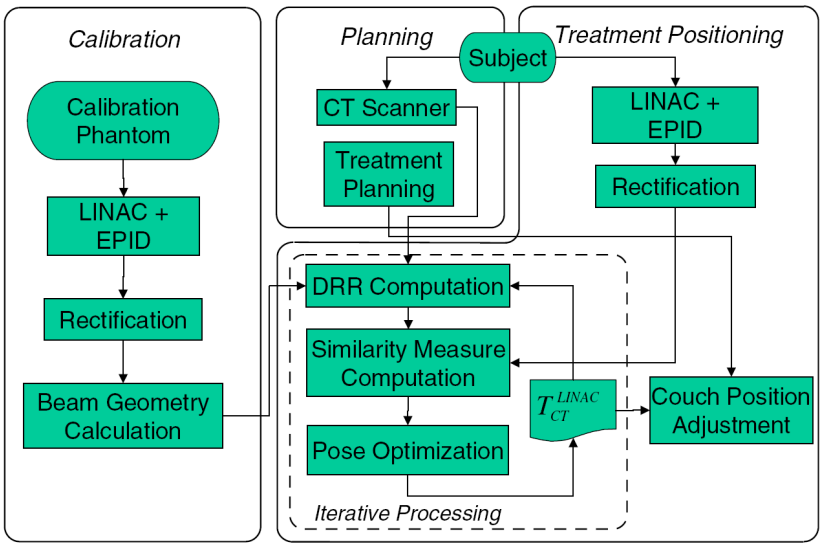


Figure 3.12: Flowchart of registration process [84].

Khamene et al. used an intensity-based method to perform 2D/3D image registration between the floating images (DRRs) and portal images (LINAC). They rendered DRRs by computing the integration of attenuation values along the ray path through the CT volume, assuming that the scattered radiation of the EPID is a small amount and can be ignored (it is about 2% for 15 MV photon treatment beams with 60 cm gap between the beam iso-center and the EPID) [105]. Moreover, they developed a professional procedure called "radiometric calibration", which aims to reduce the difference (brightness and contrast) between portal and floating images, to make the registration process between these types of images simple and possible, because portal images (LINAC) generated in MV range, on the other hand floating one (DRRs) rendered from CT volume reconstructed in kV range .

Also they rendered DRRs 256×256 from a CT volume $256 \times 256 \times 216$ in about 60 ms, by implementing a volume rendering technique to generate DRRs

as in [23]. They implemented it using the GPU, to get the advantage (high performance) of using the hardware for solving this type of problems. The authors undertook a professional comparison between various types of similarity measures. Consequently, they conclude that the local normalised correlation (LNC) is the best algorithm to measure the similarities between the portal and floating images. LNC subdivides the images into blocks then computes the correlation coefficient for each block and averaging the results to get the scalar value. By maximising the scalar value within an optimisation process, they recover six degrees of freedom and use this to register the portal and floating images. They implemented gradient and non-gradient based methods (gradient descent, powell-brent and best neighbour search) and attempted to find its effectiveness using the target registration error (TRE) value. They conclude that the type of optimisation or similarity measures used does not affect the TRE value. Figure 3.13 illustrates these results for different types of similarity measures. Finally, they performed the whole registration process in about 100 second for 256×256 single portal images, and 170 second for 256×256 stereo portal images with a 90° convergence angle.

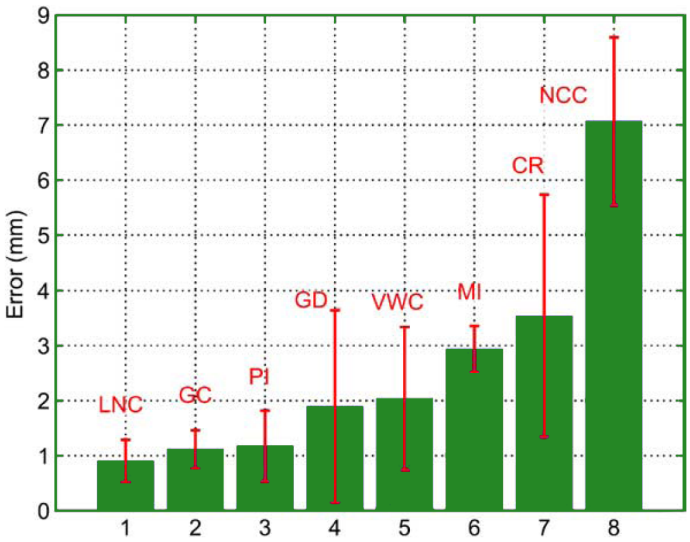


Figure 3.13: Mean and standard deviation of target registration error (TRE) for various similarity measures [84].

3.6 Summary

In this chapter we examined 2D/3D image registration and its components, focusing on DRR rendering as the main component of it. We also introduced some of the main related work on accelerating 2D/3D image registration to be able to show the prons and cons of or our methods over the other methods. Next chapter examines the performance of 2D/3D images registration using compressed CT volume, which is one of the acceleration methods we developed during our research.

Chapter 4

Performance of 2D/3D Registration using (Lossy) Compressed CT Volume

In this chapter[1] we examine the performance of the 2D/3D registration by using compressed CT volumes in order to speed up the rendering of DRR images. In Section 4.1, we describe the motivation of this approach. In Section 4.2, we give a brief introduction to the Octree data structure. In Section 4.3, we describe the method of compressing and reconstructing the CT volume and rendering DRR images. We also illustrate the performance of this method. In Section 4.4, we describe the work flow of the whole 2D/3D registration process using the lossy compressed CT volumes, then we investigate the performance of the 2D/3D registration using different compression values. Finally, in Section 4.5, we present a summary describing the results we have achieved from this study.

4.1 Motivation

2D/3D image registration requires efficient registration of many DRRs (which is the bottleneck of the registration process) and this has motivated research into

[1] *This chapter is an adapted and extended version of: "Osama Dorgham, Mark Fisher and Stephen Laycock. Performance of a 2D-3D Image Registration System using (Lossy) Compressed X-ray CT.* The Annals of the BMVA*, Volume 3, pages 1–11, 2009".*

fast algorithms and hardware acceleration [90][49][124][86][128][54][140]. Our aim in this chapter is to evaluate the registration performance of a 2D/3D registration system which uses compressed CT volume to render DRR images. An Octree compressed CT volume comprises fewer internal spaces, each containing voxels which share similar CT numbers and as such, ray casting through a volume represented as an Octree [94][14][157] is potentially computationally simpler. Little work has been published on registering 2D projections of Octree volumes and so this chapter attempts to examine the degree to which Octree compression artefacts compromise registration performance.

4.2 Octree Data Structure

An Octree is one of the hierarchical data structuring techniques used for spacial data representation of three dimensional regions [144]. The Octree was developed independently by a number of researchers [67][134][45] as a three dimensional extension of the Quadtree which has been developed with a motivation of saving storage by the aggregation of data sharing common properties within two dimensional regions. The Octree is based on the principle of recursive decomposition of three dimensional spaces. The decomposition is performed in two ways, either equally on each level (regular decomposition) or unequally, controlled by a specific input value (in our research we will use the regular decomposition, as this approach is easier than unregular decomposition and sufficient for our method) [28]. In applying Octree recursive decomposition to a three dimensional data cube the root octant which represents the entire cube domain will be divided recursively into eight octants. The process could continue until all the leaves are at the same level in the Octree hierarchy [166]. Figure 4.1 illustrates a Quadtree decomposition of a two dimensional space (we used the Quadtree in Figure 4.1 as it is easy to visualise).

Both the Quadtree and Octree find applications in different areas of computer science. Among the first credited works is that of Jackins et al. [80] who used the Octree data structure for object representation. Meagher et al. [36] used the Octree to develop algorithms for performing solid modelling operations and Yau et al. [190] used the Octree in applications of medical imaging. In computer graphics,

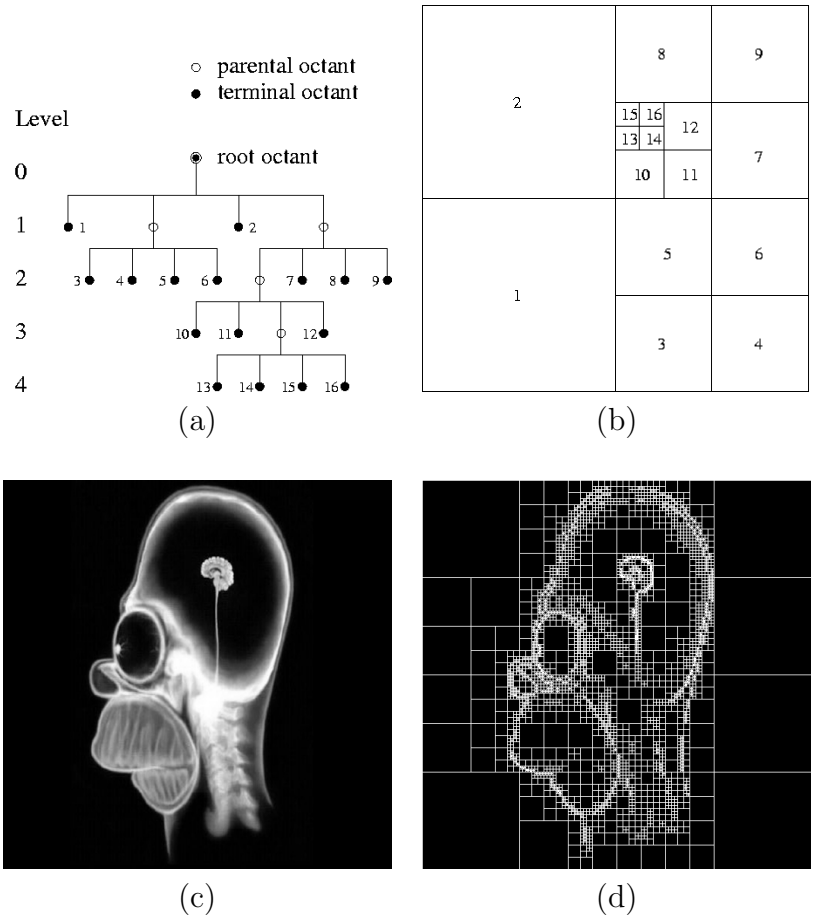


Figure 4.1: (a) Quadtree hierarchical representation, (b) Quadtree decomposition [142] with an example of (c) 2D image and (d) image Quadtree decomposition.

the Octree is an attractive data structure for volume rendering and it has been implemented in many different applications. Boada et al. [14] were the first who applied the notation of rendering Octree-compressed data by building an Octree around the main picture part (i.e. bricks), but the Octree lookup process was a bottleneck [87]. More recently, Song et al. [157] used an Octree for volume rendering (visualisation) of three dimensional medical data sets. In ray tracing applications the Octree has been applied by Knoll et al. [87] by building Octrees around voxels. The structures are then used in a fast neighbour search algorithm to return the values of cell corners when ray tracing. We used Octree as one of the most common methods of rendering three dimensional data sets [15][135][143]. In

our research an Octree compressed CT volume comprises of internal spaces, each in turn comprising voxels which share similar CT numbers is used to render DRR images. We illustrate this in the following section.

4.3 Lossy Compressed CT Volumes

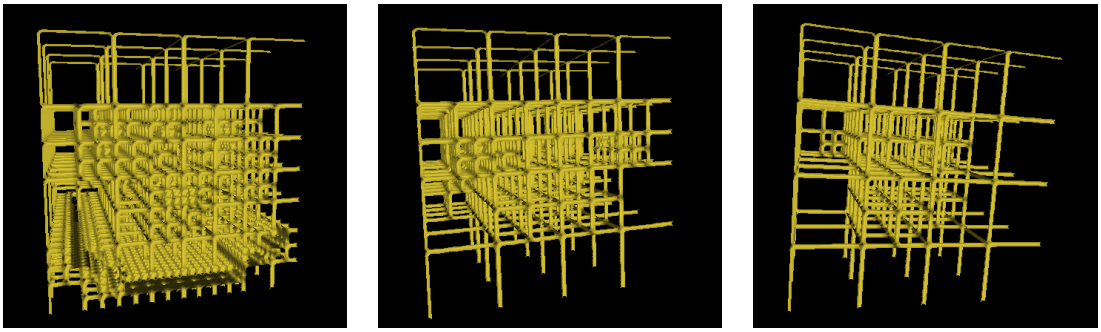
In our application we achieve compression of the CT data using the Octree by encoding the underlying voxels as a tree data structure, where each internal vertex is formed from up to eight children. Recursively we decompose the CT volume according to the CT values which share similar CT numbers as one vertex in the Octree data structure using the scheme illustrated in Section 4.3.1.

4.3.1 Scheme

The algorithm for decomposing the CT volume into an Octree runs offline. It starts by considering the internal space equal to the CT volume size and recursively splits this into eight sub-volumes (children). It is most likely that the final Octree will be an unbalanced tree (i.e. a tree which has different heights for its sub-nodes). Our method of rendering DRR images is not affected by different depths of the final tree as it is re-indexed in a hash table. However, the decision to decompose the volume is made based on a threshold T:

Require: max, min, T
  decompose(volume)
  $V = \{k : \text{k is a sub-volume}, k > 1\}$
  if $(max - min) > T$ **then**
    for all $k \in V$ **do**
      decompose(k)
    end for
    return true
  else
    return false
  end if

Where *max* and *min* represent the maximum and minimum values of voxel elements within the (sub)volume. The value of threshold T determines the degree of compression. Assuming voxels are represented by 8-bits, $0 \geq T \geq 255$, however, for generality the threshold is mapped to a parameter (P), known as the Pivot value ([0-1]). Hence $P = 0$ creates the maximum number of internal spaces. Figure 4.2 shows an example of CT volumes decomposed with different Pivot values.



P=0.4 P=0.6 P=0.8

Figure 4.2: A representation of CT volume decomposed with different Pivot values.

We store the decomposed CT volume as an ordered list of internal spaces, corresponding to leaf nodes of the Octree (starting at cell X, Y, Z = 0,0,0), each space is identified by its dimension (d) (i.e. size of the cubic space) and (I) the attenuation value within the space as illustrated in Figure 4.4. Therefore at 50% compression the memory required for uncompressed and compressed volumes is equal and at 75% compression there is a 50% saving. The results presented in Figure 4.12 suggest that at a level of 75% compression there is a modest increase in error but within clinically usable limits.

4.3.2 Rendering of DRR images

The first step in rendering the DRR images from a compressed CT volume is to reconstruct the CT volume from the corresponding Octree. The reconstructed CT volume enables DRR images to be rendered using Algorithm 1, which we described in Chapter 3, Section 3.4.5. Although, we developed an algorithm for directly rendering DRR images from Octree compressed volumes this suffered from

block artefacts as shown in Figure 4.3.

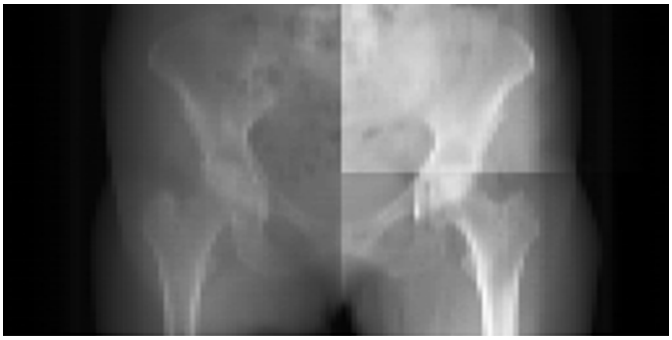


Figure 4.3: Example of DRR image shows the blocks artefact of using Octree compressed data.

Block artefacts are produced by the rendering process due to the difference in sizes of the internal spaces, which allows large spaces to contribute more to the final accumulated intensity, However, this do not affect the image registration. The decomposed compressed volume is loaded into main memory as a list data structure to save the memory space which could be improved by implementing it as a dynamic data structure, followed by the process of reconstructing (rebuilding) the compressed CT volume prior to conventionally rendering the DRR images as illustrated in Figure 4.4.

Voxel values in CT volumes are represented by a CT number quantified in Hounsfield Units (HU). The attenuation coefficient of the material comprising each voxel can be recovered by using the method that we describe in Chapter 3. Section 3.4. To render DDRs we compute the attenuation of a monoenergetic beam due to different anatomic material (e.g, bone, muscle tissue, epithelial cells, etc.) using Beer's Law [83] (more details can be found in Chapter 3). Most neighbouring voxels within the CT volumes have similar image acquisition parameters [127] and so the Octree is an appropriate technique for decomposing the CT volumes into internal spaces that share common properties. In this chapter we only address the registration performance using a conventional ray casting algorithm [153]. X-rays emanate from a point source and strike a flat panel situated behind the patient (i.e. conventional 'C' arm geometry); assumed to be lying on a flat couch. The couch or patient support system (PSS) can be rotated and translated in six degrees of freedom. The CT volume is quantised in $256 \times 256 \times 133$ 2 mm^3 voxels, and the

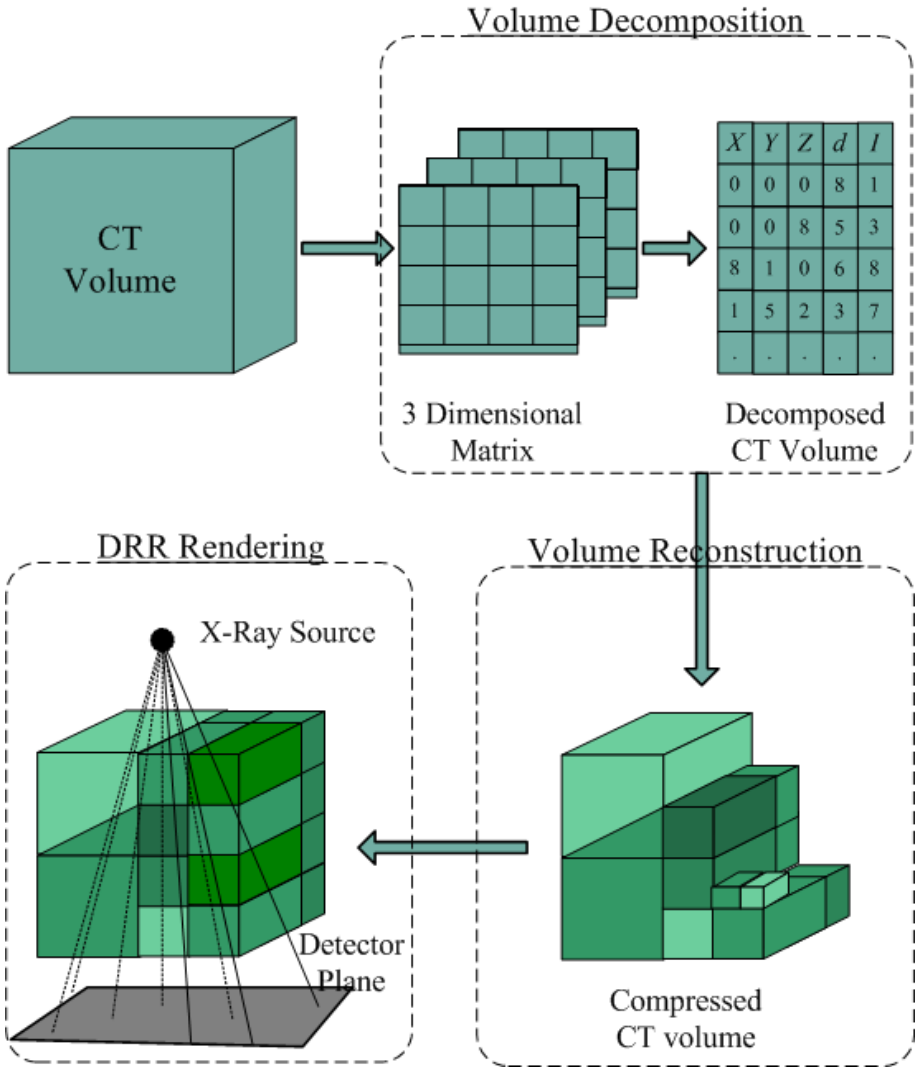


Figure 4.4: The process of rendering a DRR image from compressed CT volume.

flat panel detector models a Varian A500 amorphous silicon detector (ASD) (40 × 30 cm) operating at an effective resolution of ∼3 mm (note: the actual device resolution is a factor of 4 times better but we use low resolution DRRs to reduce computational time). The source and detector are positioned 1.5 m and 1 m from the center of the CT volume respectively. Example DRR template images from the two CT volumes are shown in Figure 4.5.

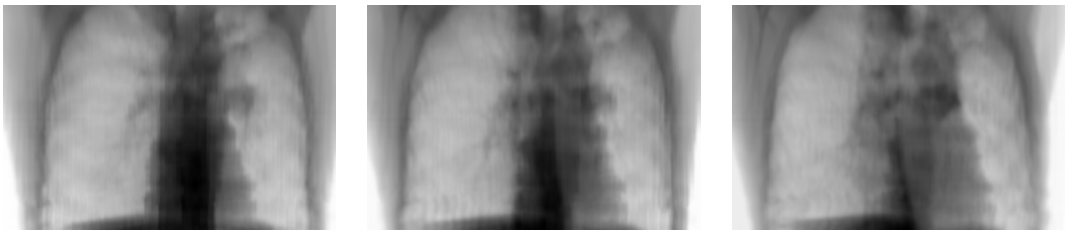
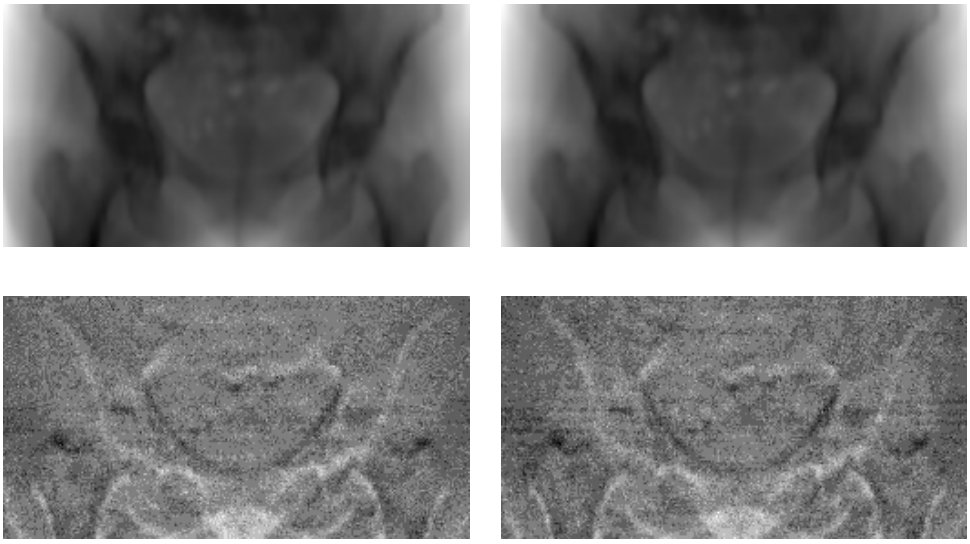
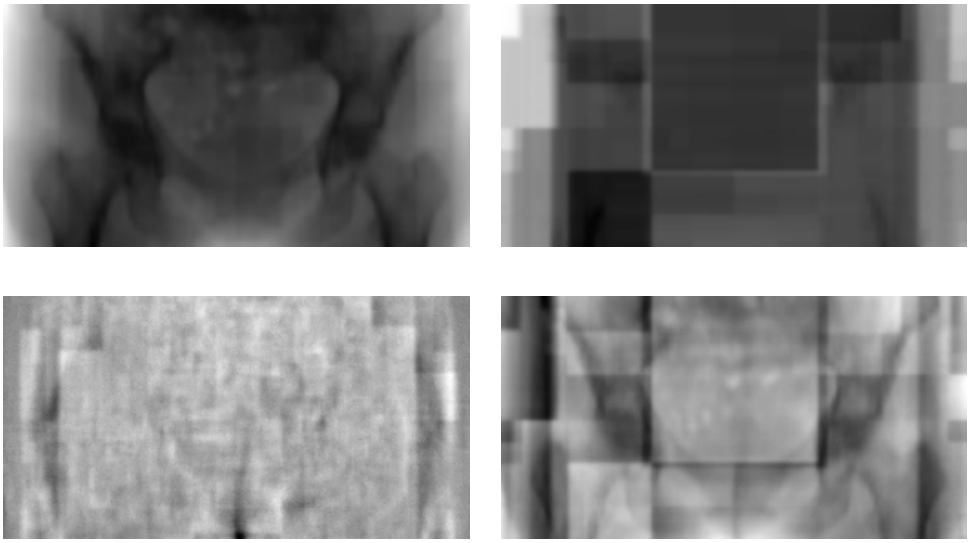


Figure 4.5: DRR template image (512×344) array rendered form lung CT volume in 12° rotation intervals around the z axis.

Compression artefacts are apparent in DRR images derived from compressed CT volumes, but at factors of compression of approximately 50% these are visually imperceivable and images remain of reasonable quality even when higher levels of compression have been applied to the CT volume (Figure 4.6). Compression artefacts are better visualised in difference images, computed by subtracting a compressed DRR image (rendered from compressed CT volume) from the original DRR image (0% compression).



(51.4%, P= 0.001) (76.6%, P= 0.01)



(92.8%, P= 0.09) (99.8%, P= 0.5)

Figure 4.6: Example of DRR images (256×133) at various compression values, and difference images formed by pixel by pixel subtraction from an uncompressed DRR (black pixel = no difference)

4.3.3 Performance

An example of DRR images reconstructed from Octree CT volume using a range of P values are shown in Figure 4.6. Table 4.1 and Figure 4.7 illustrates the relationship between P and the compression achieved with respect to a specific (pelvic) CT volume (in terms of the total number of internal spaces rendered).

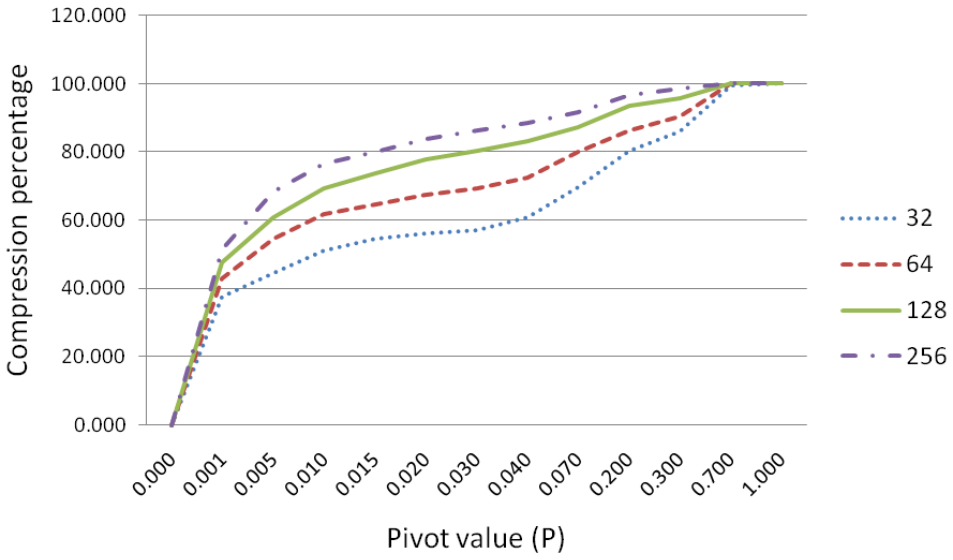


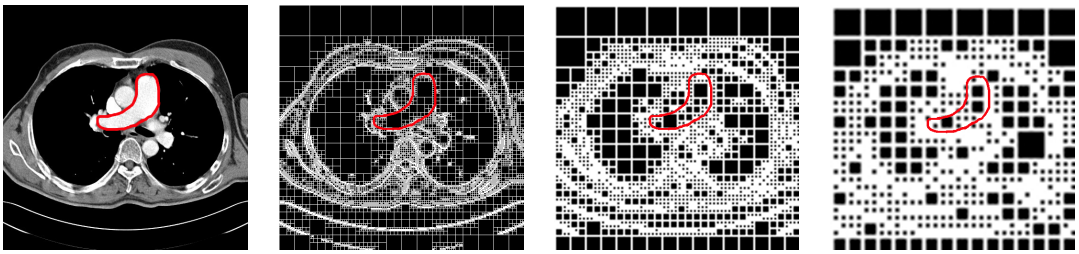
Figure 4.7: Percentage of compression in pelvic CT volume at specific P values.

The percentage compression we achieve (at the same (P) value), differs according to the size of the CT volume. Octree volumes will generate similar compression artefacts to those seen in Quadtrees, illustrated in Figure 4.8. When there are a large number of (voxels, pixels) share the same intensity values and decomposed in the same internal space, that will lead to higher percentage of compression if the internal space decomposed for smaller CT volume space. Therefore, the effect of block artefacts will be insignificant (Figure 4.8(b)) but when there are fewer, the effects will be visually apparent (e.g. Figure 4.8(d)).

The Octree decomposition process is computationally intensive. Decomposed CT volume can be generated in $O(n^3 \log_8 n)$ time, where n represents the size of CT volume, but this is not a concern because they can be precomputed. Each DRR takes 390 ms using an Intel® Core™ 2 Duo Processor T7200, 4M Cache,

	CT volume Size / Compression (%)							
P	32^3		64^3		128^3		256^3	
0	32768	0%	262144	0%	2097152	0%	16777216	0%
0.001	20469	37.53%	149248	43.07%	1097804	47.65%	8159096	51.37%
0.005	18145	44.63%	118903	54.64%	818595	60.97%	5293744	68.45%
0.01	16017	51.12%	100108	61.81%	643455	69.32%	3925720	76.60%
0.015	14904	54.52%	92457	64.73%	550565	73.75%	3342557	80.08%
0.02	14400	56.05%	85800	67.27%	464850	77.83%	2718829	83.79%
0.03	14022	57.21%	80382	69.34%	417075	80.11%	2337959	86.06%
0.04	12874	60.71%	72465	72.36%	358079	82.93%	1929607	88.50%
0.07	9948	69.64%	52592	79.94%	267485	87.25%	1423241	91.52%
0.2	6483	80.22%	35792	86.35%	141051	93.27%	556872	96.68%
0.3	4628	85.88%	25705	90.19%	93073	95.56%	249425	98.51%
0.7	162	99.51%	239	99.91%	377	99.98%	1142	99.99%
1	1	100%	1	100%	1	100%	1	100%

Table 4.1: Percentage of compression and total number of internal spaces in pelvic CT volume at specific P values (Compression value calculated as the reduction in number of spaces relative to the uncompressed spaces number i.e. Volume compression = 1 - (number of internal spaces / total number of voxels).



(a) (b) (c) (d)

Figure 4.8: Illustration of the relation between the size of CT slice and the size of decomposed spaces. Where (a) shows 512×512 CT slice before decomposition, (b) 256×256 decomposed slice, (c) 128×128 decomposed slice, and (d) 64×64 decomposed slice.

2.00 GHz to compute using the scheme which presented in Section 4.3.1 and a minimum of 15 are needed for registration, so this is a significant factor which limits the usefulness of the approach.

4.4 Registration using Lossy Compressed CT Volumes

In 2D/3D registration, patient alignment is achieved by iteratively solving an optimisation problem in six degrees of freedom. In this study we investigate the 2D/3D registration system using lossy compressed CT volumes, encoded using an Octree data structure. The process workflow of 2D/3D registration using decomposed compressed CT volume is visualised in Figure 4.9.

The other two components of the 2D/3D registration are hidden inside the registration block as shown in Figure 4.9. Similarity measurement and optimisation processes which form part of the registration are performed iteratively. This requires the DRR rendering process to find the optimum rigid transformation. In the optimisation process that we have discussed in Chapter 3, we estimated a solution by posing this as 3 independent optimisation problems in two degrees of freedom. In each case we model the similarity metric by fitting a 2^{nd} order polynomial to a sparse set of target similarity metric values. This reduces the number of time-consuming DRR images needed compared to a brute-force hill-

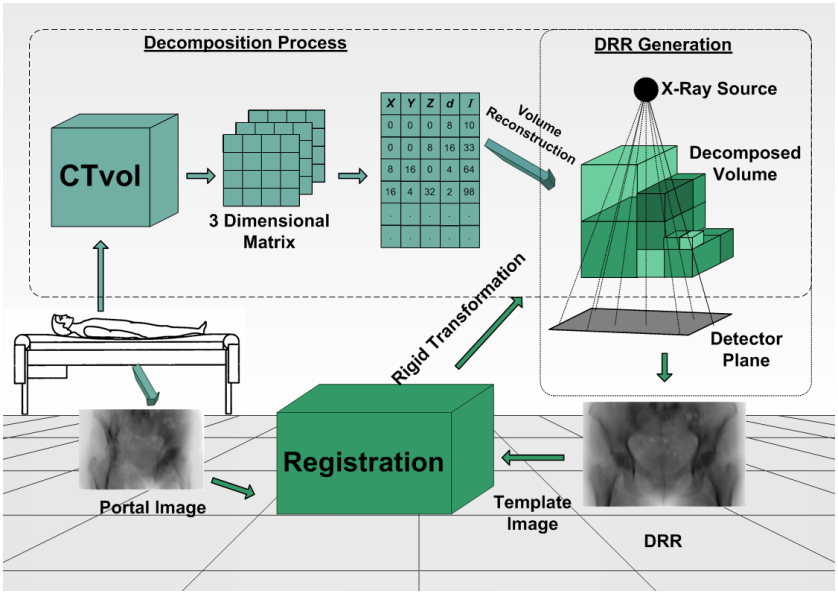


Figure 4.9: Process work flow of 2D/3D registration using decomposed compressed CT volume.

climbing algorithm and thus reduces the time required for registration. For each of the 3 optimisations the parameters (x, y, z, yaw, pitch, roll) were paired. As much as possible we choose uncorrelated pairs i.e. {x, pitch},{y, yaw},{z, roll}. For each pair we render five DRRs and compute the similarity metric for each. (Figure 4.10).

According to our optimisation strategy that we have discussed in Chapter 3, a 15 DRRs (3×5) needed to be rendered in each iteration, which is substantially fewer DRRs than and equivalent conventional best neighbour search. In the registration similarity process images are compared using normalised cross correlation (Eqn. 4.1) and the optimisation process adjusts the six parameters which controlling attitude and position of the PSS (note: other image similarity criteria were explored in Chapter 3).

$$NCC(I_{ref}, I_j) = \frac{\sum_{i=1}^{N}(A_i - \overline{A}) \cdot (B_i - \overline{B})}{\sqrt{(\sum_{i=1}^{N}(A_i - \overline{A})^2) \cdot (\sum_{i=1}^{N}(B_i - \overline{B})^2)}} \tag{4.1}$$

where A represents the reference image I_{ref} and B the reconstructed radiograph I_j. N is the total number of image pixels. At each iteration the maximum of the

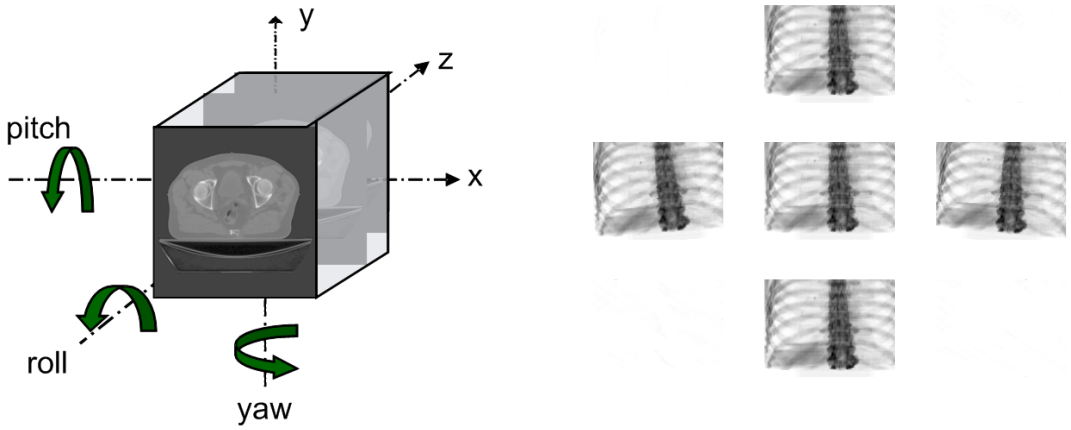


Figure 4.10: (a) six parameters of rigid transformation. (b) sparse set of DRRs needed to recover {x, pitch}

similarity metric distribution is calculated using the 2^{nd} order model, typically polynomial models for the three similarity metric surfaces as shown in Figure 4.11. In [128] the authors use a similar optimisation approach but estimate the the similarity metric distribution from a sparse set of values using a Support Vector Machine (SVM). To assess the performance of the registration with respect to compressed CT volumes reference DRR images were rendered simulating the ASD at full resolution (512×384). PSS parameters were randomly perturbed in the range ± 10 mm, ± 10 degrees. The 2D/3D registration algorithm was then used to recover the PSS parameters. The experiment was repeated 100 times for a variety of compression levels for both pelvic and lung CT volume. The target registration errors (TRE) are summarised in Table 4.2 and Figure 4.12.

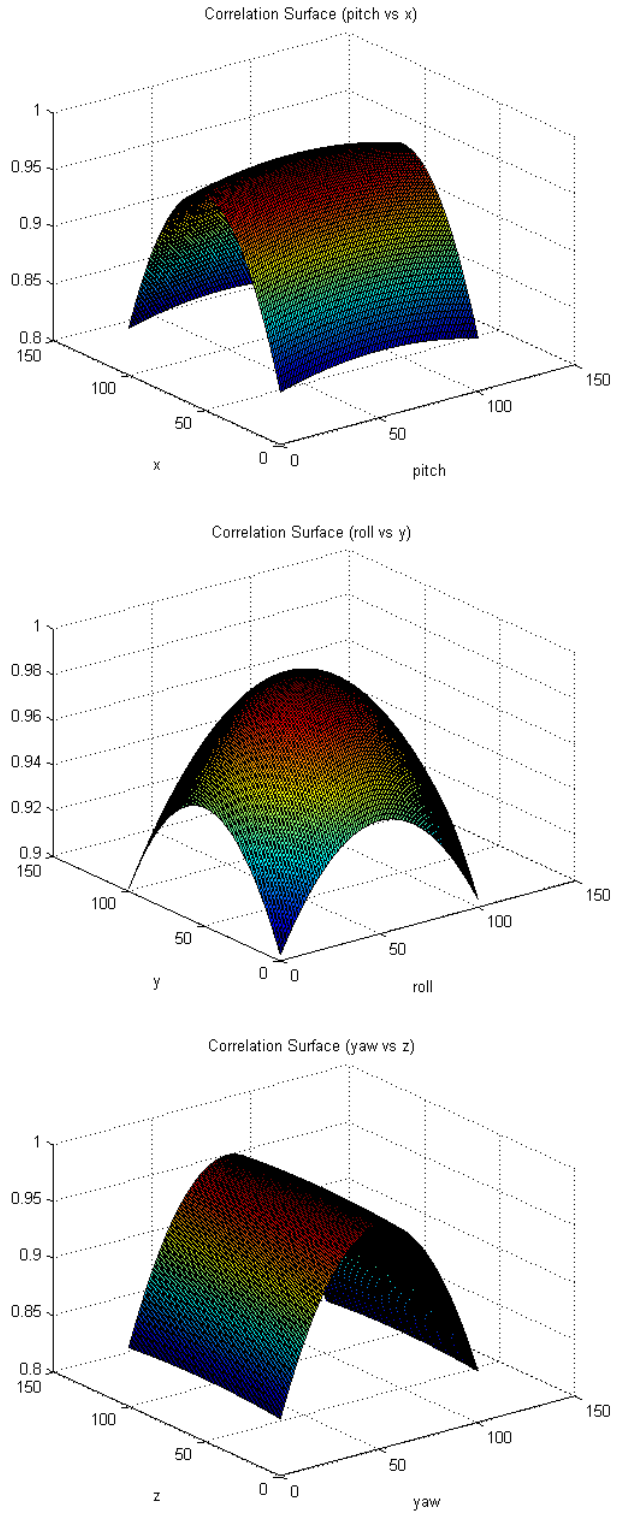


Figure 4.11: Example similarity metric surface estimates (least squares 2^{nd} order polynomial fit)

	% Compression					
DOF	0%	51%	76%	83.8%	93.1%	99.8%
$\bar{x}$	0.2000 mm	0.2250 mm	0.2400 mm	0.2550 mm	0.2560 mm	2.7390 mm
σ	0.3045 mm	0.3554 mm	0.3823 mm	0.4136 mm	0.4162 mm	1.7823 mm
$\bar{y}$	1.3920 mm	1.7310 mm	1.8260 mm	1.9010 mm	1.8880 mm	8.6250 mm
σ	1.8225 mm	1.9154 mm	2.0550 mm	2.1472 mm	2.2637 mm	10.3380 mm
$\bar{z}$	0.2060 mm	0.6560 mm	0.6720 mm	0.6750 mm	0.6120 mm	2.7240 mm
σ	0.2768 mm	0.3227 mm	0.3502 mm	0.3603 mm	0.3701 mm	3.3388 mm
$\bar{p}$	0.3870°	0.5010°	0.5000°	0.4910°	0.4630°	12.6150°
σ	0.3765°	0.4050°	0.4298°	0.4361°	0.4482°	5.1920°
$\bar{w}$	0.0550°	0.0580°	0.0590°	0.0580°	0.0610°	1.2230°
σ	0.0769°	0.0808°	0.0805°	0.0782°	0.0802°	1.4262°
$\bar{r}$	0.1760°	0.1370°	0.1410°	0.1400°	0.1190°	6.4390°
σ	0.2254°	0.1745°	0.1817°	0.1787°	0.1611°	6.6575°

Table 4.2: Mean and standard deviation target registration error (TRE)(p = pitch, w = yaw, r = roll).

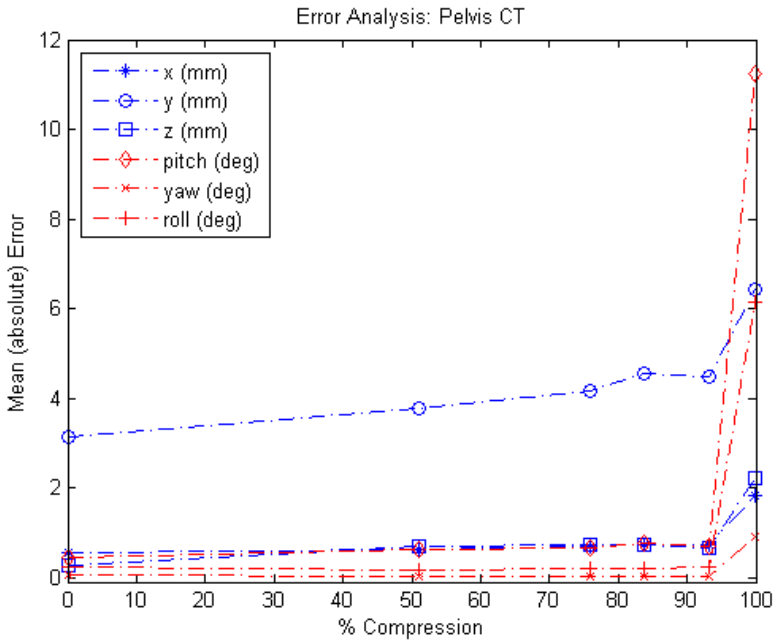


Figure 4.12: Mean target registration error (TRE) (p = pitch, w = yaw, r = roll).

4.5 Summary

The results show that 2D/3D registration can recover PSS {x,y,z} translation of up to ±10 mm. with sub-voxel accuracy (i.e $\ll$ 2 mm.) and angular {pitch, yaw, roll} rotations up to ±10 degrees with an accuracy of better than 1 degree. Furthermore, the performance does not degrade significantly when Octree compression is used at levels up to about 90%. The errors in recovered y translations are larger than in x and z since the X-ray fan beam is nearly parallel ($\sim 4.5°$) and so the geometry is insensitive to adjustments in the height of the couch. In practice, the height of the couch (i.e. patient) much less likely to change compared to x, y translation and p, w, r rotation of the anatomy and so this is not seen as a significant problem. That the 2D/3D registration scheme (0% CT volume) delivers similar TRE to other published studies [84][49] is not significant, particularly as the evaluation uses simulated fluoroscopy images which are of higher quality. But it is surprising that this performance does not suffer through the use of a polynomial estimate of the similarity metric distribution, and in particular the

accuracy is well within 3 mm required for most radiotherapy patient setup applications. The fact that the performance remains acceptable even at relatively high CT volume compression rates is much more interesting. In our knowledge this is the first study to demonstrate that compressed volumetric data might be used within a 2D/3D registration framework. Besides the obvious memory saving these data structures afford they appear to offer efficiencies to ray casting approaches. Further ray casting approaches that exploit concurrency in CPU and GPU hardware to speed up the rendering of DRR images are illustrated in Chapter 5 and 6.

Chapter 5

Parallelised Accelerated Rendering of DRR Images

In this chapter[1] we examine an approach to speeding up the rendering of DRR images which parallelises the casting of rays used in rendering DRR images. In Section 5.1, we describe the motivation for this approach, in Section 5.2 we give a brief introduction to parallel processing and describe some of popular languages and frameworks. In Section 5.3, we explore high speed rendering of DRR images from CT data volumes by parallel processing on multiple CPU cores and we describe the (API) library we used to develop our software solution. In Section 5.4, we summarise the findings of this chapter.

5.1 Motivation

The motivation of this work can be explained by the importance of performing the registration process in a quick way as we illustrated in Chapter 3. We address this by parallelising the process of rendering DRR images to be able to render DRR images in a quick way to perform faster registration process.

[1] *This chapter is an adapted and extended version of: "Osama Dorgham, Mark Fisher and Stephen Laycock. Accelerated Generation of Digitally Reconstructed Radiographs using Parallel Processing.* In Proc. Medical Image Understanding and Analysis 2009*, Jul. 14-15 2009, pages 239-243".*

5.2 Parallel Processing

Parallelising a task is not a new concept for mankind and many ingenious systems have been developed over the years. Figure 5.1 illustrates a system of ploughing the land developed a long time ago to perform the job faster and more easily.

Figure 5.1: Picture illustrating the concept of parallelising the tasks from long time ago [62].

In computing the concept is the same but the task and the tools are different. In computing we can define parallel processing as the simultaneous use of multiple processor cores to speed up program execution [172]. Different languages and APIs have been introduced during the last few decades to address the problem of concurrent programming. So in this section we will give a brief introduction to some popular languages of parallel processing such as Occam, Pascal-FC, ADA and OpenMP which represent one of the most recent application programming interface (API) for parallel processing.

5.2.1 Occam

Occam is a parallel programming language designed to express concurrent program execution on a network of programming components. Occam was developed in 1982 at Inmos Limited (Bristol, England) with a main goal of keeping the language simple. With this in mind it is named after William of Occam, a thirteenth century philosopher who had the philosophical principle of “keep things simple”. Occam enables an application to be executed as a collection of concurrent processes,

where each process can communicate with other processes through channels. From the Occam model, Inmos developed the transputer to support their concurrent model. Transputers are 32-bit microprossors which can be interconnected via serial communication channels to form flexible parallel machines. Hence, the language and the hardware are both designed in such a way that concurrent processes can be executed on one transputer or divided over many transputers [18][68].

5.2.2 Pascal-FC

Pascal-FC (Functional Concurrent Pascal) is a programming language developed for use in the teaching of concurrent programming. Pascal-FC was developed in 1988 as an extension of Wirth's Pascal to be a simple and compact language for concurrent programming. Communication and synchronisation primitives were added to support semaphores, CSP/Occam type rendezvous and monitors [35].

5.2.3 Ada

The Ada programming language is a high level programming language mainly designed for the production of massive embedded real-time systems [52][88]. Ada was the result of the most extensive and expensive design effort ever undertaken for a programming language [129]. It was designed by the United States Department of Defense (DoD) from 1977 to 1983. The name of the language refers to Countess Ada Lovelace (1815-1852) who is considered to be the world's first programmer [162] (A painting of her is presented in Figure 5.2).

Ada is not just designed for computationally intensive mathematical problems, it has been designed to support various models of parallelism. Furthermore, Ada is designed to support a concurrent programming model which was subsequently used for developing programmes for parallel shared memory machines [4]. In the early years after Ada's development it began by supporting basic vector computations, employing arrays of processors for Single Instruction Multiple Data (SIMD) machines. Later it also supported a shared memory model of parallelism known as the Multiple Instruction Multiple Data (MIMD) parallel computing model using Message Passing [101].

Figure 5.2: Augusta Ada King, Countess of Lovelace, an English scientist born in 1815, lived with her mother after her parents separation. In 1842, Charles Babbage asked her to translate a paper about the analytical engines, although he asked her to add her notes to the translation. After one year of an extensive work she did the job which has been published in The Ladies' Diary and Taylor's Scientific Memories. In 1953, after more than hundred years of her death, Ada's notes have been republished and recognised to be the earliest description to the computer and software [129][43][81].

5.2.4 OpenMP

A few years ago, hardware companies started to compete by developing a new generation of powerful personal computers based on multi-core technology, which supports the simultaneous execution of threads in a shared memory system (parallel execution). But to take advantage of this improvement, it is necessary to deploy a parallel programming software model based on the shared memory architecture. The software solution was the birth of OpenMP in October 1997, as an effective parallel programming model able to support many multi-core systems with shared memory and distributed shared memory. OpenMP is becoming the standard for parallelised applications, as it has been adopted by major computer manufacturers and software development companies including Compaq, IBM, SGI, Sun, Intel,

Hewllett-Packard, Kuck & Associates (KAI) and the U.S. Department of Energy ASCI program. Together these formed the Architecture Review Board (ARB). OpenMP is not a programming language, it is an application programming interface (API) which supports programmes written in a sequential programming languages like Fortran, C or C++ enabling them to execute applications on different cores using threads on a shared memory. In sequential programming languages (e.g C++) parallelisation is performed by using a parallel programming model which could be implemented as an API (OpenMP), to provide a bridge between the sequential languages and the programmer to develop parallelised programmes [64]. OpenMP fulfilled the goals of ARB members to create a user-friendly API, since it is used to parallelise the code and leave other programming issues to the compiler. Also, it is a widely adopted API, so applications developed using OpenMP can be executed on many different platforms. Moreover, OpenMP was created to survive for the future, where the fast development on shared-memory parallel computers (SMPs) and multithreading hardware needs an efficient shared-memory standard like OpenMP [136][12].

5.3 Fast Rendering of DRR Images by Parallel Processing

This section explores parallel processing on the CPU for DRR rendering. Parallel processing has made a tremendous impact on a variety areas of computer science (e.g. graphics, simulation, security and image processing). In our case, we render DRR images by casting rays using more than one processor in order to obtain faster results, which means we are now be able to process more than one ray concurrently. Multithreaded techniques are well suited to multi-core CPUs found in many modern PCs and we have obtained a significant performance increase in the rendering of DRR images with this approach. Our objective is to increase the speed of DRR rendering using parallel processing. To test the approach, we developed and implemented the following algorithm (Algorithm 2) in C++ using parallel programming model (OpenMP library).

This algorithm is designed not only to describe how the DRR image will be

Algorithm 2 Parallel processing of DRR rendering using OpenMP

```
1: OpenMP set num of threads(2^n)
2: OpenMP start parallel "for" loop statement
3: with two "counters" (+:absorptionSum) and (+:count)
4: also with "private" variables (j,t0,stratTime,endTime,intersectionPoint,absorption)
5: for all i such that 0 ≤ i ≤ imgDimX do
6:    count = i × imgDimY
7:    for all j such that 0 ≤ j ≤ imgDimY do
8:       x_ray = x_rays[count]
9:       absorptionSum = 0
10:      for all t0 such that stratTime ≤ t0≤ endTime do
11:         if CTimg.intersection(x_ray) then
12:            intersectionPoint = x_ray.getPosition(t0)
13:            absorption = CTimg[CTimg.offset(intersectionPoint)]
14:            absorptionSum = absorptionSum + absorption
15:         end if
16:      end for
17:      drr.setAbsorption(absorptionSum )
18:      count = count +1
19:   end for
20: end for
```

rendered, but also, how the work (rendering process) can be distributed and decomposed across the multiple cores. Considering Algorithm 1 (Section 3.4.5), we parallelised the DRR rendering process by assigning each vertical line of rays to an individual core as a separate thread using the variable "count" as illustrated in Figure 5.3.

Algorithm 2 also implements the special algorithm of box ray intersection points [8]. The aim of this implementation is to quickly calculate the coordinates of the intersection of the ray and the CT volume. A point based algorithm [9][66][74] is implemented to speed up the rendering process by sampling the intersection points within the CT volume. Different types of rendering algorithm could be implemented, such as ray casting [47], splatting [186] or shear-warping [176], but generally they exhibit a higher time consumption of $O(N^3)$ time complexity, compared to the point based algorithm of $O(N^2)$ time complexity [9]. The execution time of the parallel DRR algorithm differs according to the number of cores. Nor-

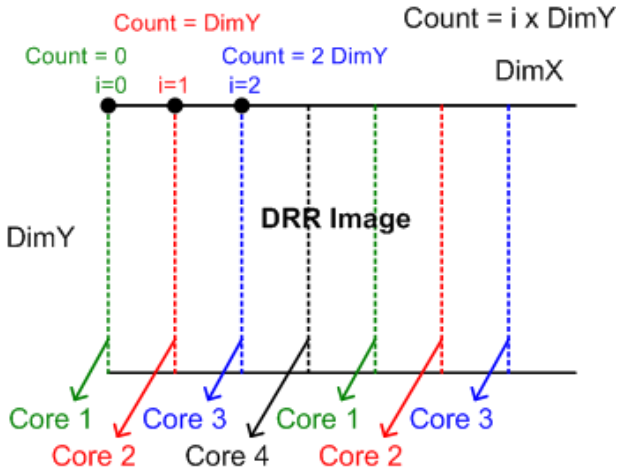


Figure 5.3: Illustration for the parallelisation process using algorithm 2.

mally a large number increases the number of parallel threads which reduces the rendering time of the DRR images, by casting multiple rays simultaneously. Also, the size of the CT volume affects the total time of the DRR rendering and the number of sampled points inside the CT volume per ray.

Writing our algorithm using OpenMP does not offer algorithmic capabilities that are not already available in C or C++. So the main reason to program in OpenMP is performance [136]. An obvious concept to achieving improved performance for parallel implementation is to parallelise a sufficiently large portion of code (i.e. large portion of code means; part of the code represents most of program code's lines or/and the highest complexity part of the code). But in some cases the performance of the application can be constrained by the serial portion of the program (i.e. portion of the program can not be parallelised). So, according to Amdahl's law [136], if F is the parallelised portion of the code and S_e is the speed up achieved in the parallel portion, the overall speed up S will be:

$$S = 1/[(1 - F) + F/(S_e)]$$

Therefore, we can find the maximum improvement of the DRR rendering process when it is parallelised, as illustrated in Figure 5.4.

In Figure 5.4 we show results using Amdahl's law for different parallelised

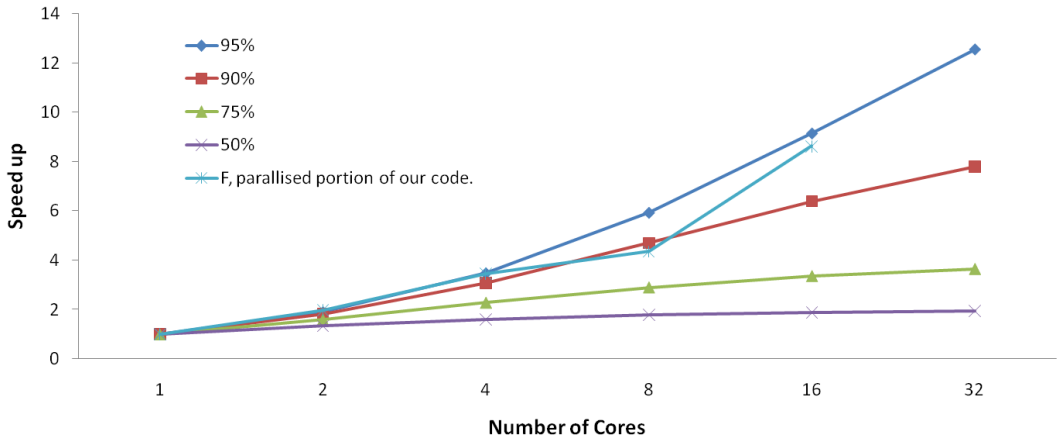


Figure 5.4: The relation between number of processes, parallelised portion of code and the speed up gained according to Amdahl's law and our results of speeding up the DRR rendering.

portion of code at 95%, 90%, 75% and 50% (as in Tables 5.1- 5.4), together with our parallelised portion of code for DRR rendering (according to the results in Table 5.5). Our results in Table 5.5, match the 90% parallel portion curve of Amdahl's law with an overall speed up of more than three times in comparison between a single and quad cores and more than eight times between single and hexa cores. DRR rendering requires a large number of rays to render high-quality images. DRR images rendered at full resolution require $p \times q$ rays, where p and q are the dimensions of the required image. The acceleration we achieved varied according to the size of the CT volume data and the number of processors. The growth and availability of multi-core technology provides a low cost computing platform to speed up the rendering of the DRR images.

F	N	**Execution Time**	**Speed Up**
		$T = (1 - F) + F/N$	$1/T$
0.95	1	1	1
0.95	2	0.525	1.904762
0.95	4	0.2875	3.478261
0.95	8	0.16875	5.925926
0.95	16	0.109375	9.142857
0.95	32	0.079688	12.54902
0.95	64	0.064844	15.42169
0.95	128	0.057422	17.41497
0.95	256	0.053711	18.61818
0.95	512	0.051855	19.28437
0.95	1024	0.050928	19.63567

Table 5.1: The speed up results for different number of cores at 95% parallelised portion of code according to Amdahl's law, where F is the fraction of parallelism, N is the number of processors and T is the time.

F	N	**Execution Time**	**Speed Up**
		$T = (1 - F) + F/N$	$1/T$
0.9	1	1	1
0.9	2	0.55	1.818182
0.9	4	0.325	3.076923
0.9	8	0.2125	4.705882
0.9	16	0.15625	6.4
0.9	32	0.128125	7.804878
0.9	64	0.114063	8.767123
0.9	128	0.107031	9.343066
0.9	256	0.103516	9.660377
0.9	512	0.101758	9.827255
0.9	1024	0.100879	9.912875

Table 5.2: The speed up results for different number of cores at 90% parallelised portion of code according to Amdahl's law, where F is the fraction of parallelism, N is the number of processors and T is the time.

F	N	**Execution Time**	**Speed Up**
		$T = (1 - F) + F/N$	$1/T$
0.75	1	1	1
0.75	2	0.625	1.6
0.75	4	0.4375	2.285714
0.75	8	0.34375	2.909091
0.75	16	0.296875	3.368421
0.75	32	0.273438	3.657143
0.75	64	0.261719	3.820896
0.75	128	0.255859	3.908397
0.75	256	0.25293	3.953668
0.75	512	0.251465	3.976699
0.75	1024	0.250732	3.988315

Table 5.3: The speed up results for different number of cores at 75% parallelised portion of code according to Amdahl's law, where F is the fraction of parallelism, N is the number of processors and T is the time.

F	N	**Execution Time**	**Speed Up**
		$T = (1 - F) + F/N$	$1/T$
0.5	1	1	1
0.5	2	0.75	1.333333
0.5	4	0.625	1.6
0.5	8	0.5625	1.777778
0.5	16	0.53125	1.882353
0.5	32	0.515625	1.939394
0.5	64	0.507813	1.969231
0.5	128	0.503906	1.984496
0.5	256	0.501953	1.992218
0.5	512	0.500977	1.996101
0.5	1024	0.500488	1.998049

Table 5.4: The speed up results for different number of cores at 50% parallelised portion of code according to Amdahl's law, where F is the fraction of parallelism, N is the number of processors and T is the time.

Full Resolution DRR (Best Possible Resolution)	Number of Cores							
	Single		Dual		Quad		Octal	
Image Direction / CT Volume Size	0°	90°	0°	90°	0°	90°	0°	90°
$128 \times 128 \times 66$ (pelvis)	116 ms	109 ms	81 ms	57 ms	55 ms	49 ms	34 ms	26 ms
$128 \times 128 \times 86$ (lung)	139 ms	137 ms	77 ms	72 ms	66 ms	64 ms	34 ms	33 ms
$256 \times 256 \times 133$ (pelvis)	1003 ms	857 ms	765 ms	489 ms	459 ms	388 ms	231 ms	196 ms
$256 \times 256 \times 172$ (lung)	1301 ms	1103 ms	1035 ms	697 ms	552 ms	450 ms	292 ms	255 ms
$512 \times 512 \times 267$ (pelvis)	9013 ms	6658 ms	4280 ms	4053 ms	2986 ms	2749 ms	1666 ms	1456 ms
$512 \times 512 \times 344$ (lung)	9821 ms	8668 ms	7425 ms	5994 ms	3955 ms	3431 ms	2142 ms	1856 ms

Table 5.5: Results in milliseconds of running the parallelised algorithm for rendering full resolution DRR images using different numbers of cores and different sizes of CT volumes (pelvis and lung), DRR images were rendered from y direction at 0° and 90° using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz.

5.4 Summary

In this chapter, we developed a method of accelerating the process of 2D/3D image registration by parallelising the DRR images. This was motivated by a need to improve the speed of this process while maintaining sufficient clinical accuracy. The speed of rendering DRR images depends mainly on use of parallel processing (OpenMP) on multiple CPU cores and the resolution of DRR images. The growth and availability of multi-core technology provides a low cost computing platform to speed up the generation of the DRR images. Validity and durability of our method will not stop while the improvement is going on to the processing capabilities. An important consideration is that using a parallel processing approach on the CPU does not require any pre-processing steps unlike our previous method of DRR rendering we presented in Chapter 4 which required the CT volume to be compressed or methods presented in [145][141][90]. Additionally, we are able to render DRRs from multiple view points over six degrees of freedom (x, y, z, yaw, pitch, roll), without any restriction or limitation on the camera position ($\Delta r = 1^{\circ}, \Delta t = 1mm$). On the other hand, DRR images could be rendered using a reduced number of rays, by interpolating the missing values or reduced number of sampling points. The rendering of reduced resolution DRR images will be discussed in detail in (Chapter 7, Section 7.2) and their use in 2D/3D registration for kV reference images is assessed in terms of TRE.

Chapter 6

GPU Acceleration for Digitally Reconstructed Radiographs

Recent advances in GPU programming languages provide developers with a convenient way of developing applications which can be executed on the CPU and GPU interchangeably. In this chapter we introduce a way of accelerating the image registration process by developing a hybrid system which executes on the CPU and GPU. In Section 6.1 we describe the motivation of running the registration process as a hybrid system. In Section 6.2 we present the work related to the rendering of DRR images using the GPU and discuss the efficiency of it in order to give the reader an easy way of comparing our results to the previous results of speeding up the registration process. In Section 6.3 we give a brief introduction to GPU programming and to the APIs, libraries and programming languages which used in GPU programming for parallel data processing. In Section 6.4 we use CUDA to develop an accurate solution of rendering the DRR images. Then we compare the GPU methods of rendering the DRR images to each other. Finally, in Section 6.5 we describe the final structure of the hybrid registration system and evaluate the performance of it.

6.1 Motivation

Considering the improvement of the results we achieved by parallelising the process of rendering DRR images using OpenMP for the CPU implementation, we decided to investigate if further benefit of multi-core technology could be achieved by executing our DRR rendering algorithm on the Graphical Processing Unit (GPU) using a parallel computing architecture called the Compute Unified Device Architecture (CUDA) programming environment. Furthermore, a comparison between the two implementations motivates us to investigate the current performance and the future trends for both technologies in order to find the optimal solution for the registration problem.

6.2 Related Work

Development of powerful multi-core GPUs was initially motivated by the games industry and they have been used widely in various applications of computer graphics. However, image processing and computer vision have also benefited from this technology as many techniques require computation over many pixels, either for still images or video (e.g. edge detection or face recognition) [69]. Recently, manufacturers of GPUs have targeted a wide range of computer intensive applications through the development of APIs such as CUDA. In this section we first introduce different methods that have been used to implement the registration process on the GPU and the speed gained to render the DRR images. The next part focuses on the speed and the latest results of using CUDA in image registration, specifically in the DRR rendering. As expected, most of the current research has paid attention to the rendering of DRRs on the GPU devices and leaves the other components of the registration process to be performed on the CPU device. The reason for this choice is that the DRR rendering is the most computationally expensive process in any rigid registration method.

One of the earliest GPU-based acceleration studies for 2D/3D image registration was introduced by LaRose [90] in his PhD thesis for iterative 2D/3D registration. The technique relies on an OpenGL extension implemented on a GeForce 3 graphics card. Using the accumulating algorithm which runs on graphics hard-

ware, they were able to accelerate the rendering of the DRR images to render them at a size of 512×512 pixels from CT volume of size $256 \times 256 \times 256$ voxels. The rendering was achieved at a speed of approximately 14 Hz (71 ms per DRR image). During the last 10 years there have been significant enhancements in the GPU industry. This is clearly seen if we compare LaRose's results to the following recent results.

- From the literature, Grabner et al. [51] developed their method of Automatic Differentiation (AD) 2D/3D image registration to be executed on a hybrid CPU/GPU system using the Cg language (Note: Cg has been developed by NVidia to support parallel GPU implementations, more information presented in Section 6.4). In their research they discuss the design and implementation of their algorithm on the GPU and the restrictions and conflicts that arise due to the requirements for reverse mode AD. In addition, they present a method which can register a CT data set of $512 \times 512 \times 288$ voxels with three reference X-ray images of 512×512 pixels in 11.8 seconds on a Geforce 8800 GTX NVidia card.

- Ino et al. [76] accelerated the process of 2D/3D image registration using a General Purpose computing on Graphics Processing Unit (GPGPU) approach. They used OpenGL to implement their method and perform the DRR rendering, gradient image rendering and normalised cross correlation (NCC) computation on the GPU. Their experimental results show that they rendered the DRR image from $300 \times 300 \times 48$ voxels in 17.1 ms using Pentium 4 3.4 GHz with GeForce 7800 GTX. They compared their results to a cluster based method and showed that their method was faster than a 32 node cluster of PCs with Pentium 3 1 GHz CPU. According to this study a DRR image of size 256×256 could be rendered in 66.4 ms using CT volume of size $256 \times 256 \times 256$.

- Lu et al. [98] used CUDA to implement a technique they developed to accelerate the rendering of DRR images. From their results they reported a real time reconstruction process within 3 seconds for a $128 \times 128 \times 128$ voxel CT volume from 80 128×128 pixels projection, without compromising the

image quality. According to their results this means that they need more than 37 ms to render each DRR image using Core2 Quad Q6600 CPU (2.4 GHz), 4 GB memory and an NVidia 8800 GTX GPU with 768 MB RAM and CUDA 2.0. According to this study a DRR image of size 256×256 could be rendered in 296 ms using CT volume of size $256 \times 256 \times 256$.

- Mori et al. [111] developed a GPU-based (CUDA) application to render DRR images. They performed a comparison between the CPU and GPU based applications of rendering DRR images. Their CPU-based application requires about 8500 ms to render a single DRR image, but using the GPU based application it requires 2.8 minutes to render 784 DRR images which means it takes approximately 214 ms per DRR image from a CT volume of size $512 \times 512 \times 256$ voxels using an NVidia 8800 GTS GPU. According to this study a DRR image of size 256×256 could be rendered in 53.5 ms using CT volume of size $256 \times 256 \times 256$.

- Bethune et al. [13] studied DRRs as an individual process in the medical image processing field. They showed its importance in image registration, illustrated by applications in planning of orthopedic surgery and viewing intra-articular features which are not visible in surface-shaded CT images. Their technique used the hardware capabilities of the GPU to detect empty regions (air, soft tissues, etc.) in the CT volume, then to avoided rendering these regions in order to accelerate the DRR rendering process. They did not explain if the process of detecting empty regions was a pre-processing or not, but from the description of their technique it seems it was performed as a pre-process operation. Experimental results showed that the accelerated and unaccelerated rendered DRRs were identical, and could be rendered by the GPU with a speed up factor of up to 2.9, without any loss in the DRRs quality. A DRR of size 512×512 pixels rendered from a CT volume of size $512 \times 512 \times 128$ voxels with frame rate of 5.4 frames per second from the *entire* volume and with frame rate of 22.5 frames per second ($\sim$44 ms per DRR) from the *boxed* volume (accelerated method) using 2.6 GHz Pentium PC with an Nvidia 5900 graphics card containing 256 MB of video memory. According to this study a DRR image of size 256×256 could be rendered

in 22 ms using CT volume of size $256 \times 256 \times 256$.

- Finally, the latest work has been done by Spork in his thesis [160]. Spork presents a GPU-based high speed rendering algorithm for DRR images. He also did a comparison between the GPU-based Wobbled splat rendering algorithm [186] and his GPU-based ray casting implementation. He compared the quality and the performance of the algorithms. Results show that he can render DRR images from CT volume of size $256 \times 196 \times 196$ voxels in about 13 ms with opacity threshold of 70 (i.e. rendering very few voxels) using NVidia Quadro FX 570M GPU with 512 MB memory, (where we required 11 ms to render a DRR image using larger CT volume of size $256 \times 256 \times 133$ voxels with bigger amount of voxels as the threshold is 50% of the CT volume volxels and using NVidia 8800 GTX GPU). According to this study a DRR image of size 256×256 could be rendered in 22 ms using CT volume of size $256 \times 256 \times 256$.

6.3 General Purpose Computing on Graphics Processing Unit (GPGPU)

Nowadays, Graphics Processing Units (GPUs) are relatively cheap, powerful and available hardware components, which can be used to perform intensive calculations, especially in graphics and game computing. The last decade of hardware performance developments and future trends show that GPU-based computation is significantly faster than the CPU-based computation as illustrated in Figure 6.1.

GPUs can be used to do general purpose scientific and engineering computing, the general use of GPU applications have been called General Purpose Computing on Graphics Processing Unit (GPGPU) [59]. The purpose of this section is to give a brief introduction to the GPGPU programming and to the APIs, libraries and programming languages which are support and used in GPU programming for parallel data processing.

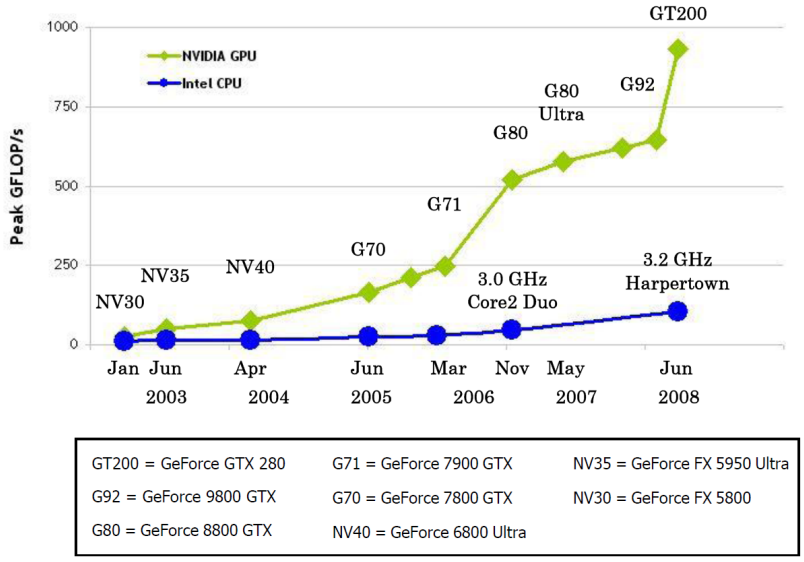


Figure 6.1: Development of the CPUs (Intel) and GPUs (NVidia) from 2003 to 2008 measured by the Giga FLoating point Operations Per Second (GFLOPS/s) [118].

6.3.1 OpenGL

OpenGL (Open Graphics Library) is an application programming interface (API) established as an industry standard by Silicon Graphics and other graphics hardware companies in 1993. OpenGL is designed to access graphics hardware at the lowest level to provide applications with high performance. OpenGL specification and conformance tests are controlled by the ARB (for more information see Section 5.3). In June 2003, 10 years after releasing the first specification of OpenGL the ARB approved the inclusion of a high-level graphics programming language called the OpenGL Shading Language as an extension to OpenGL [133]. Normally GPUs have at least two programmable processors called the vertex and fragment processor. The GPU vertex processor undertakes tasks related to the vertex shaders such as colour computation, vertex position transformation or computing values for lighting per pixel. The following example of OpenGL code would send to the vertex processor a vertex position and colour for each vertex.

```
glBegin(GL_POINTS)
    glColor3f(r,g,b);
```

```
        glVertex3d(x,y,z);
glEnd();
```

On the other hand, the fragment processor operates the tasks related to the fragment shaders such as computing texture coordinates per pixel, fog computation or computing normals for lighting per pixel. The fragment processors inputs are interpolated values that are a result of the previous stage in the pipeline such as vertex position, colour, etc. The main advantage of the fragment processor is that it can access texture memory, combining texture values in random ways. A fragment shader can read multiple values from multiple or single textures. Also results from one texture access can be used conditionally to perform another texture access, so that ray casting algorithms can be processed in a fragment processor as a fragment shader [133].

6.3.2 Cg Language

C for Graphics (Cg) is a shading language developed for GPU programming, introduced by NVidia in close collaboration with Microsoft in December 2002 [98]. Cg is derived from C but includes some specific adjustment to be compatible with the features of graphics hardware; so Cg code looks like C code with the same function calls, declarations and most data types. Cg is a shading language compatible with OpenGL and DirectX and is used mainly for vertex and fragment shaders, which allows programmers to write programs for both the vertex processor and the fragment processor [42][116]. Finally, the main reason of developing the Cg language is to use a high-level programming language for hardware, rather than the low-level language such as Assembly language which is painful to use and presents serious difficulties to the effective use of hardware [116]. So, we can imagine Cg as a language that has been developed to render objects easily using programmable graphics hardware (i.e. a language for graphics hardware programming).

6.3.3 Compute Unified Device Architecture (CUDA)

CUDA™ was introduced by NVidia in November 2006, as a general purpose parallel computing architecture. The CUDA software environment is designed to support familiar high-level programming languages like C to make it easy for programmers to learn. Also, it is designed to support application programming interfaces, such as OpenCL (Open Computing Language is a trademark of Apple Inc., used under license by Khronos [72]), CUDA FORTRAN and DirectCompute [117]. CUDA is a scalable parallel programming model, where programs written using CUDA could be executed on any number of processors without recompilation. The core of CUDA consists of three abstractions: *thread groups*, *shared memory*, and *barrier synchronisation*, which provide different levels of data parallelism and thread parallelism. This different level of parallelism would make it possible to decompose the programming problem into sub-problems which can be solved in parallel using blocks of threads, then into finer pieces that can be solved cooperatively in parallel by all threads within the block which will preserve CUDA scalability [118]. In CUDA thread groups abstraction, threads can be divided into multiple equal-shape thread blocks which are organised into a one-dimensional, two-dimensional, or three-dimensional grid of thread blocks illustrated in Figure 6.2.

Threads are indexed by defining it as a three component vector, so that threads can be identified using a one dimensional index, two dimensional index (D_x, D_y) with ID of $(x + yD_x)$, and a three dimensional index (D_x, D_y, D_z) with ID of $(x + yD_x + D_xD_y)$. In CUDA shared memory abstraction, threads can access data from multiple memory spaces during their execution. All threads have access to the same global memory, but each thread has its private local memory. Also, each thread block has its shared memory which is visible to all threads of the block as illustrated in Figure 6.2. Memory resources are limited for each core. Therefore, the number of threads per block is limited as well, since all threads of a block are expected to be located on the same processor core. Threads of a block are able to communicate with each other by reading and writing per-block shared memory at a synchronisation barrier. In CUDA barrier synchronisation abstraction, a barrier for threads of a block guarantee synchronisation and concurrent execu-

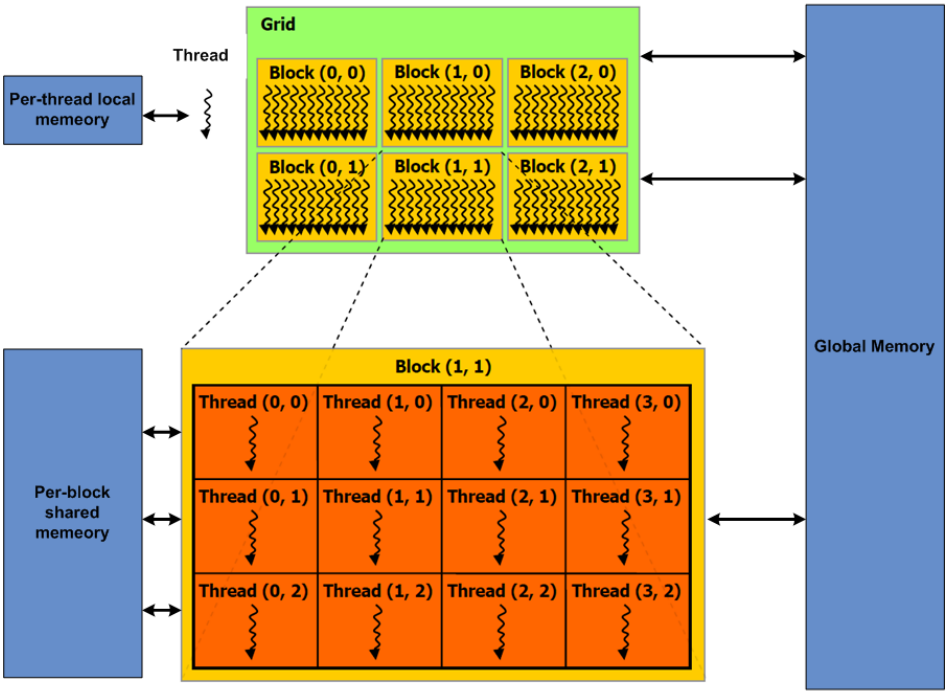


Figure 6.2: Grid of blocks of threads that execute in parallel [118].

tion, where no thread sharing the barrier can proceed until all threads reach the barrier [118][114].

6.4 CUDA-based Rendering for DRR Images

The main reason for us to choose CUDA to implement our algorithms on the GPU is that CUDA supports any application to be processed on the GPU, unlike Cg for example which is mainly useful for graphics applications. Other reasons to choose CUDA are that we do not need to worry about the physical structure of the GPUs when it is an NVidia product and it is relatively easy to find NVidia products that are suitable for applications in the health industry.

6.4.1 An Accurate Method of Rendering DRR images using CUDA

Considering the high processing capabilities of the GPUs, we decided to improve the quality of DRR images by implementing another ray casting algorithm. This algorithm is capable of rendering accurate DRR images using a voxel traversal structure [7]. We did not initially investigate this approach as it needs more computation compared to the sampling algorithm [9] which we have implemented and introduced in a previous chapter. This method of rendering fast DRR images samples the voxels inside the CT volume along the X-rays path ($\vec{u} + t\ \vec{v}$, $0 \leq t \leq 1$), by finding the entry (In) and exit (Out) intersection points (t_{in}, t_{out}) for each ray and the CT volume. We then increment by Δt from t_{in} to find all the intersection points (voxels) along the X-ray path (for more details please return to Section 3.4). The sampling method would be ideal if the X-rays are parallel since the sampling interval could be set to deliver accurate results. But in our application X-rays are cast as a cone beam and this introduces inaccuracies as samples may not be taken in every voxel and the distance traversed across each voxel will vary (depending on the angle of incidence). Our problem lies in the fact that some X-rays which are not parallel to the direction of the voxels inside the CT volume will result in intersecting some voxels more than once, as we illustrate in Figure 6.3.

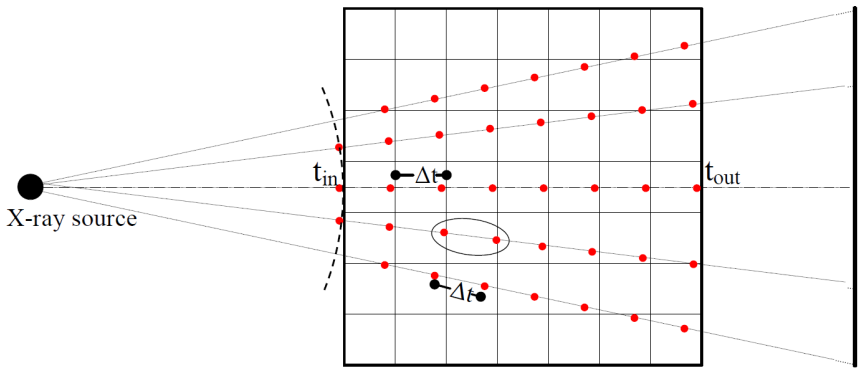


Figure 6.3: 2D illustration of the artefact of intersecting X-rays (cone beam) with CT volume.

Incrementing by Δt, as a fixed value along the path of the X-rays will result in a small error of sampling some voxels more than once. Rendering DRR images using the cone bean and fixed sampling suffers from this artefact. To eliminate the artefact we implement a fast and simple incremental grid algorithm which was developed by Amonatides et al. [7] to sample each voxel only once despite the direction of the X-rays, as illustrated in Figure 6.4.

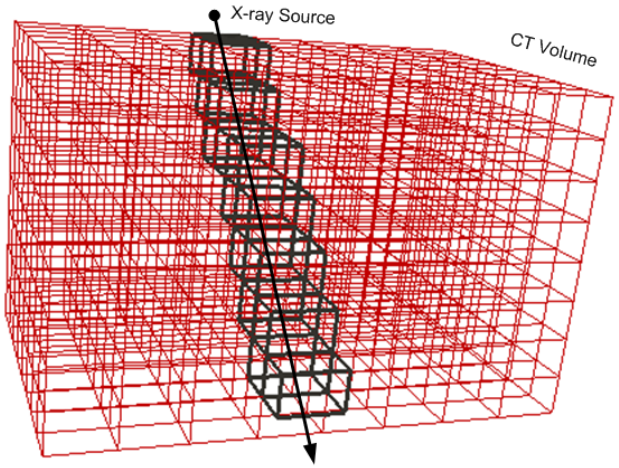


Figure 6.4: 3D illustration for the incremental grid algorithm which ensures voxels are only sampled once.

This traversal algorithm consists of two phases, initialisation and incremental traversal. The initialisation phase calculates the parameter values which will cause the ray to move into a new slice of the grid. The variables $tMaxX$, $tMaxY$ and $tMaxZ$ are used to store the parameter values for movements in the x, y and z axes respectively. Δx, Δy and Δz are used to store the parameter values required to move along the ray such that a distance of one cell width, one cell high and one cell deep is traversed. Then the incremental traversal phase can determine all the voxels that intersect with the X-rays by incrementing the exact movement along each ray. The algorithm starts from the first box the ray cuts and looks at the smallest parameter value stored in $tMaxX$, $tMaxY$ and $tMaxZ$ to determine the next box which must be chosen. The variables are updated and the algorithm repeats until the next selected box no longer resides in the voxel grid. This algorithm is one of the fastest for ray tracing, as it requires very few floating

point operations and it ensures each voxel can be sampled only once. Abstract pseudo code for the DRR algorithm using CUDA is illustrated in Algorithm 3.

Algorithm 3 CUDA-based rendering of DRR images

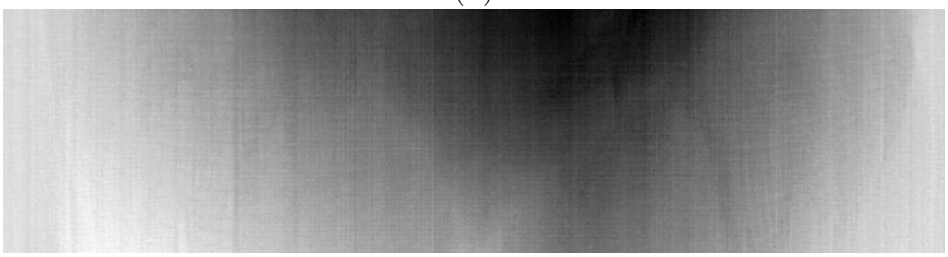
```
1: x_raysList.clear();
2: x_raySource.set();
3: for all i such that 0 ≤ i ≤ imgDimX do
4:    for all j such that 0 ≤ j ≤ imgDimZ do
5:       index ← i × imgDimZ + j
6:       host_x_raysDirection[index × 3].x() ← detectorPlane[index].x();
7:       host_x_raysDirection[index × 3 + 1].y() ← detectorPlane[index].y();
8:       host_x_raysDirection[index × 3 + 2].z() ← detectorPlane[index].z();
9:    end for
10: end for
11: CUDA_MemoryCopyHostToDevice(device_x_raysDirection, host_x_raysDirection);
12: CT.startIntersectionPoint.set();
13: CT.endIntersectionPoint.set();
14: CUDA_calculateAbsorption(device_DRR, device_x_raysDirection, x_raySource,
    x_raysList, CT.startIntersectionPoint, CT.endIntersectionPoint)
15: {
16: d_index ← imgDimY × imgDimZ + imgDimX;
17: x_ray_absorption ← calculate_x_ray_absorption_using_fast_voxel_traversal_alg();
18: device_DRR[d_index] ← x_ray_absorption;
19: }
20: CUDA_MemoryCopyDeviceToHost(host_DRR, device_DRR);
21: host_DRR.display();
```

As illustrated in the previous algorithm, CUDA supports transfer of CT data between the main memory (host memory) and the local hardware memory (device memory). This operation is performed once during the DRR rendering process (line 11 in Algorithm 3), which eliminates the time required to move the data for each ray calculation. In Figure 6.5, we are introducing examples of DRR images rendered from pelvis CT volume data using the accurate and the sampling methods of DRR rendering which we described in Chapter 3, Section 3.4.5.

In order to compare the quality of DRR images rendered using the two methods, we performed a quantitative comparison between the sampled DRR image and the accurate one by calculating the peak signal-to-noise ratio (PSNR).

(a)

(b)



(c)

Figure 6.5: Sample of DRR images rendered from pelvis CT volume data using (a) Amonatides' et al. [7] algorithm (very accurate), (b) sampling algorithm which we described in Chapter 3, Section 3.4.5 and (c) difference image formed by pixel by pixel subtraction between the two images (black pixel = no difference).

$$PSNR = 20 \times \log_{10}\left(\frac{R}{\sqrt{MSE}}\right)$$

Where R is the maximum pixel value and MSE is the mean square error. Results show that DRR images rendered using both of the methods are very similar in quality as the PSNR ratio between the DRR images is above 36 dB . On the other hand, the GPU-based sampling method of rendering DRR images is faster than an accurate rendering of DRR images by 1.8 times (accurate rendering 68 ms ÷ sampling method 38 ms, using NVidia GeForce 8800 GT of rendering

DRR images from CT volume of size $256 \times 256 \times 133$ voxels (pelvis)). Therefore, we decided to keep the sampling method as the reference method for all further steps in our research.

6.4.2 Performance Evaluation of GPU-based Rendering of DRR Images

In the previous chapter we illustrated our enhanced CPU-based method of rendering DRR images. In this section we will evaluate the performance of our GPU-based rendering method by comparing it to previous results we got from our CPU-based rendering method. In this comparison we will focus on the *image quality* and *rendering time* of DRR images as the main measures of performance.

Rendering Time (Speed)

Comparing the speed between the GPU and CPU is not an "apple to apple" comparison due the difference in the architecture between them. To enable a comparison we implement the same algorithm for rendering DRR images on the GPU and on the CPU. Results show that the speed is significantly reduced for the GPU-based method when compared to the CPU-based one, as illustrated in Figure 6.6.

Table 6.1 and 6.2 show the timing results of rendering different size and resolution DRR images using a GPU Nvidia GeForce 8800 GTX and Nvidia GeForce GTX 580, respectively. The GeForce 8800 GTX contains 128 streaming processor cores running at a frequency of 575 MHz and has a total memory of 768 MB. The GeForce GTX 580 contains 512 streaming processor cores running at a frequency of 1544 MHz and has a total memory of 1536 MB.

According to the related work which we introduced in an earlier part of this chapter, we believe that Mori et al. [111] presents the fastest method of rendering full resolution DRR images. But comparing our results to their results, we believe that we recorded the fastest results of rendering full resolution DRR images. Using our method we are able to render a DRR image from $512 \times 512 \times 267$ CT volume in about 184 ms using NVidia GeForce 8800 GTX where they rendered a DRR

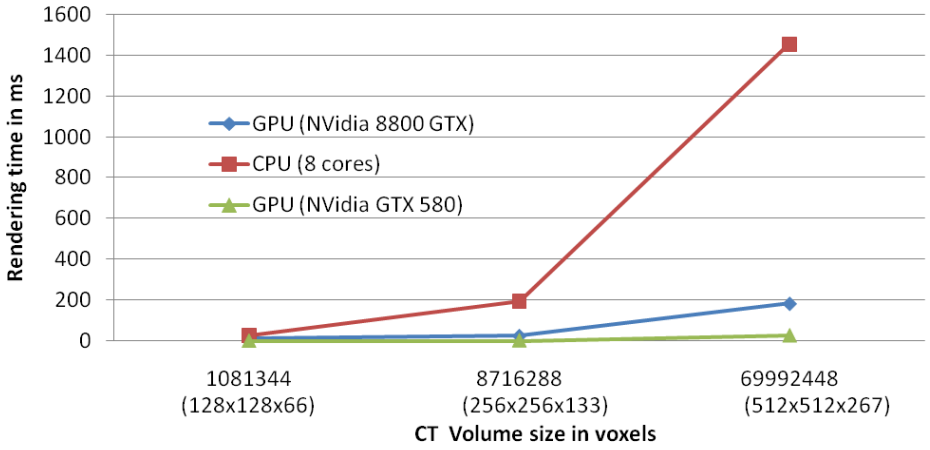


Figure 6.6: Comparison of the speed of the GPU-based method versus the CPU-based method of rendering DRR images using different sizes of CT volumes. CPU-based DRR images were rendered using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz. GPU-based DRR images were rendered using NVidia GeForce 8800 GTX.

	DRR Images Resolution		
CT volume size	**Full resolution**	**50% sampling**	**25% sampling**
$64 \times 64 \times 64$ (pelvis)	11 ms	11 ms	11 ms
$128 \times 128 \times 66$ (pelvis)	11 ms	11 ms	11 ms
$128 \times 128 \times 86$ (lung)	11 ms	11 ms	11 ms
$256 \times 256 \times 133$ (pelvis)	24 ms	11 ms	11 ms
$256 \times 256 \times 172$ (lung)	24 ms	11 ms	11 ms
$512 \times 512 \times 267$ (pelvis)	184 ms	104 ms	51 ms
$512 \times 512 \times 344$(lung)	237 ms	118 ms	64 ms

Table 6.1: Results in milliseconds of running the GPU-based rendering algorithm of reduced resolution DRR images using different CT volumes (pelvis and lung). DRR images were rendered with reduction in the sampled points by 50% and 75% of the FR-DRR image using NVidia GeForce 8800 GTX.

image in about 214 ms using NVidia 8800 GTS. But our CT volume is larger than their CT volume by 5%, also their DRR images are surrounded by a black area rendered as an artefact of rays not intersecting the CT volume (according to Mori)

	DRR Images Resolution		
CT volume size	**Full resolution**	**50% sampling**	**25% sampling**
$64 \times 64 \times 64$ (pelvis)	< 1 ms	< 1 ms	< 1 ms
$128 \times 128 \times 66$ (pelvis)	1 ms	< 1 ms	< 1 ms
$128 \times 128 \times 86$ (lung)	1 ms	< 1 ms	< 1 ms
$256 \times 256 \times 133$ (pelvis)	2 ms	1.5 ms	1 ms
$256 \times 256 \times 172$ (lung)	3 ms	2.5 ms	1 ms
$512 \times 512 \times 267$ (pelvis)	27 ms	13 ms	8 ms
$512 \times 512 \times 344$(lung)	35 ms	20 ms	10 ms

Table 6.2: Results in milliseconds of running the GPU-based rendering algorithm of reduced resolution DRR images using different CT volumes (pelvis and lung). DRR images were rendered with reduction in the sampled points by 50% and 75% of the FR-DRR image using NVidia GeForce GTX 580.

and we think this area represents about 34% of the total size of the DRR image. Where our DRR images rendered using rays intersect all the CT volume which means we perform more calculations to find all the intersection points for each ray. Moreover, the resulting times of rendering the DRR images which are presented in Table 6.1 and Table 6.2 include the calculation times of the rays direction, the transformation operation, and presenting the DRR image on the screen. From Table 6.1 results show that small DRR images (128×66 and 128×86 pixels) which are rendered using small CT volumes ($128 \times 128 \times 66$ and $128 \times 128 \times 86$ voxels) will require about 11 ms to render each DRR image, which we believe is not the real amount of time required to render the DRR image itself but it is the time for projecting the image onto the screen and performing the other internal operations in the DRR rendering process like copying memory from the host to the device, calculating ray directions and detector plane transformations, etc. whereas within a registration system there is no need to view the DRR image during the registration process as all the calculations can be performed on the graphics card after rendering the DRR images. The structure of the whole registration process is presented in the next section.

The results show that the GPU-based rendering (NVidia GeForce 8800 GTX)

required approximately 24 ms to render a full resolution DRR image from a $256 \times 256 \times 133$ CT volume, which gives a speed up ratio of 8 times over the results of a CPU-based rendering (using OpenMP with 8 cores) introduced in the previous chapter. More results for the speed up ratio between the GPU-based rendering and CPU-based rendering of DRR images are presented in Table 6.3.

	Speed up ratio		**PSNR**
CT volume size	GPU/CPU(1 core)	GPU/CPU(8 cores)	**ratio**
$128 \times 128 \times 66$ (pelvis)	10	2.4	∞
$128 \times 128 \times 86$ (lung)	12.5	3	∞
$256 \times 256 \times 133$ (pelvis)	35.7	8.1	∞
$256 \times 256 \times 172$ (lung)	46	10.6	∞
$512 \times 512 \times 267$ (pelvis)	36.1	8	∞
$512 \times 512 \times 344$(lung)	36.6	7.8	∞

Table 6.3: Speed up ratio for the GPU-based method over the CPU-based method, and the PSNR ratio between DRR images rendered using both methods. CPU-based DRR images were rendered using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz. GPU-based DRR images were rendered using a NVidia GeForce 8800 GTX .

Image Quality

We evaluate the quality of the DRR images by calculating the PSNR ratio between the DRR images which were rendered using the GPU-based and CPU-based methods . Results from Table 6.3 shows that there is no difference at all between the DRR images as the PSNR ratio $\equiv \infty$ ($MSE \equiv 0$). Although, Mori et al. [111] reported a difference in the quality of DRR images which were rendered using GPU-based and CPU-based methods in single and double precision respectively. In our method we did not face this problem as we are not performing any interpolation operations to render DRR images and we are able to run our method using any NVidia GPU products.

6.5 Hybrid Approach of 2D/3D Registration

According to the difference in the capabilities between the CPU and GPU, and the difference in the complexity of the applications, we found that the best way to achieve the optimum performance for the 2D/3D image registration problem is to have a hybrid CPU/GPU based system that is able to get the maximum benefit from both types of processor depending on the registration process and its internal sub processes (optimisation, similarity measure and DRR rendering). For more details about 2D/3D image registration process please refer to Chapter 3. In this section we will describe the *structure* and *performance* of the 2D/3D image registration process using a hybrid system which partially executes on the CPU and partially executes on the GPU.

6.5.1 The Structure of the Registration Using a Hybrid Approach

We proposed a structure for the registration process according to the complexity of its internal processes (components) and according to the capabilities of the CPU and GPU. To evaluate the complexity of internal processes we give the registration process a mathematical description:

$$f(n) = \alpha \times (n^3 + n^2) + \zeta$$

Where α indicates the number of required transformation operations (translation and rotation) done by the *optimisation* process, n represents a dimension of the CT volume, and ζ is a constant representing the total number of support operations like computing the direction of X-rays, transforming the detector plane, etc. From the previous equation we can say that the registration process is required to render α DRR images of complexity $\mathcal{O}(n^3)$ and similarity measure operations of complexity $\mathcal{O}(n^2)$, so that the final complexity of registration process will be $\mathcal{O}(n^3)$. The high complexity of the DRR rendering process returns to the nature of the rendering process as a graphics problem (huge number of rays need to intersect all the voxels inside the CT volume). In spite of the high complexity of

this problem, an important characteristic can be exploited, namely that the ray intersections are not related to each other. Therefore, each ray can be processed individually and randomly from the other rays. We can take advantage of this characteristic by processing each ray individually using the GPU to get the benefit of the huge number of processors (128 processor using NVidia GeForce 8800 GTX) available to process the ray intersections in parallel.

On the other hand, whether the similarity measure process can be implemented as a CPU-based process or GPU-based process depends mainly on the way the DRR rendering process is implemented. In the first case, implementing the similarity measure process as a CPU-based process, requires the DRR images to be copied back from the device memory (GPU memory) to the host memory (CPU memory). This could lead to a synchronisation problem on the GPU device as it is busy all the time rendering a DRR image and so it needs a slot of time to return back the DRR image to the host memory to be evaluated by the similarity measure process. According to the previous mathematical description of the registration process we know that the similarity measure process is not a high complexity process and normally does not require a long time to be executed using the CPU, so for example it requires about 2 ms to measure the similarity between two images of size 512×267 pixels using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz. Whereas transferring the DRR images from the device memory to the host memory requires almost the same time. The first case of implementing the similarity measure process as a CPU-based process is illustrated in Figure 6.7.

In the second case, implementing the similarity measure as a GPU-based process does not require any data transfer between the device and the host memory as the DRR images are stored on the device memory. But time is still needed to perform the similarity measurement process on the GPU. However, there will not be a delay or synchronisation problem as all the required data is stored on the GPU. This requires device memory to store a DRR image of size 512×267 pixels $\equiv 4\ (float) \times 512 \times 267 \equiv 546816\ byte \equiv 0.52$ MB. These days, reasonable graphics cards can have 512 MB memory or more, so the memory required for storing a DRR image is not a problem. The second case of implementing the similarity measure process as GPU-based process is illustrated in Figure 6.8.

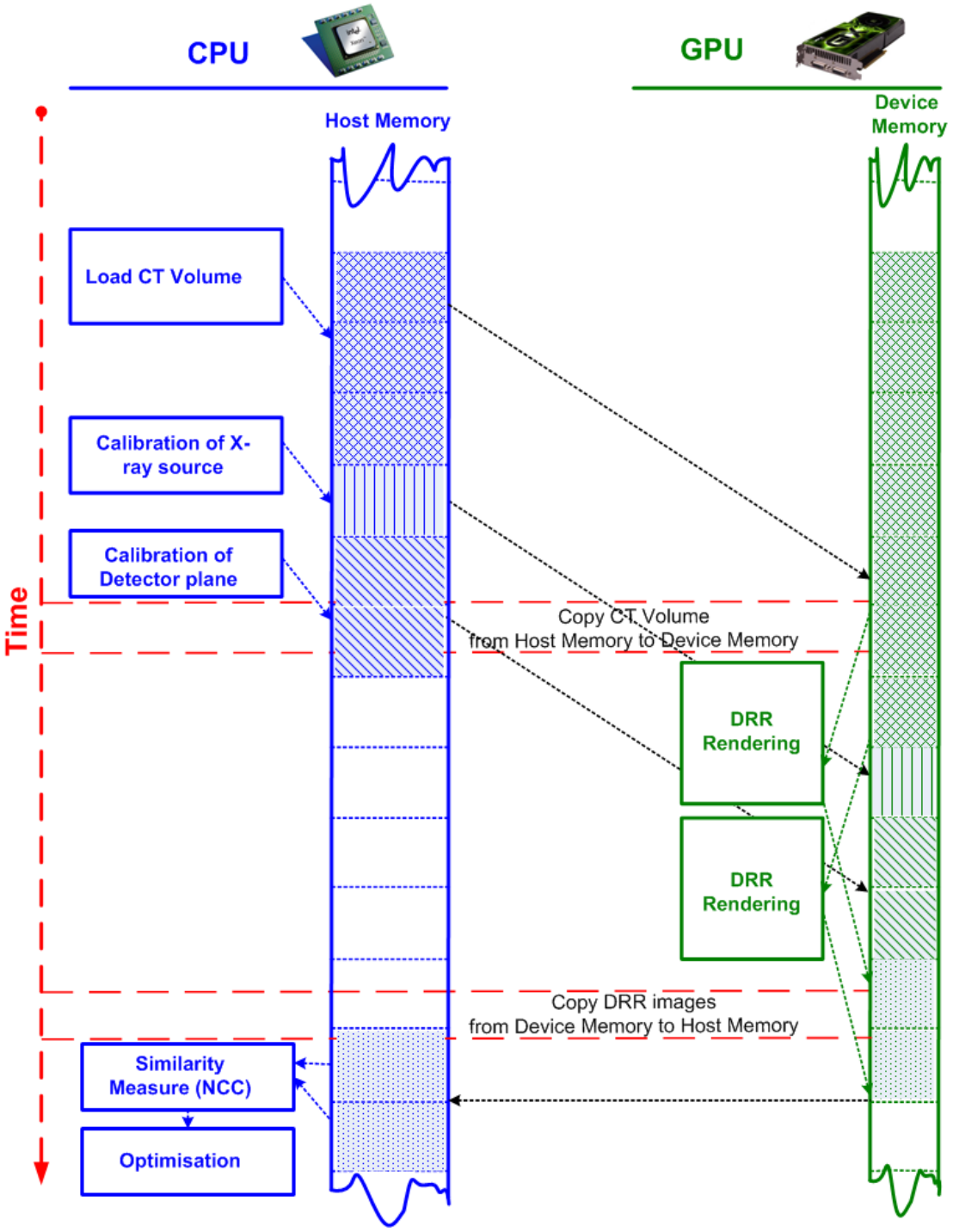


Figure 6.7: A single iteration of the registration process, where the DRR rendering is implemented as a GPU-based rendering process and the similarity measure process is implemented as a CPU-based process.

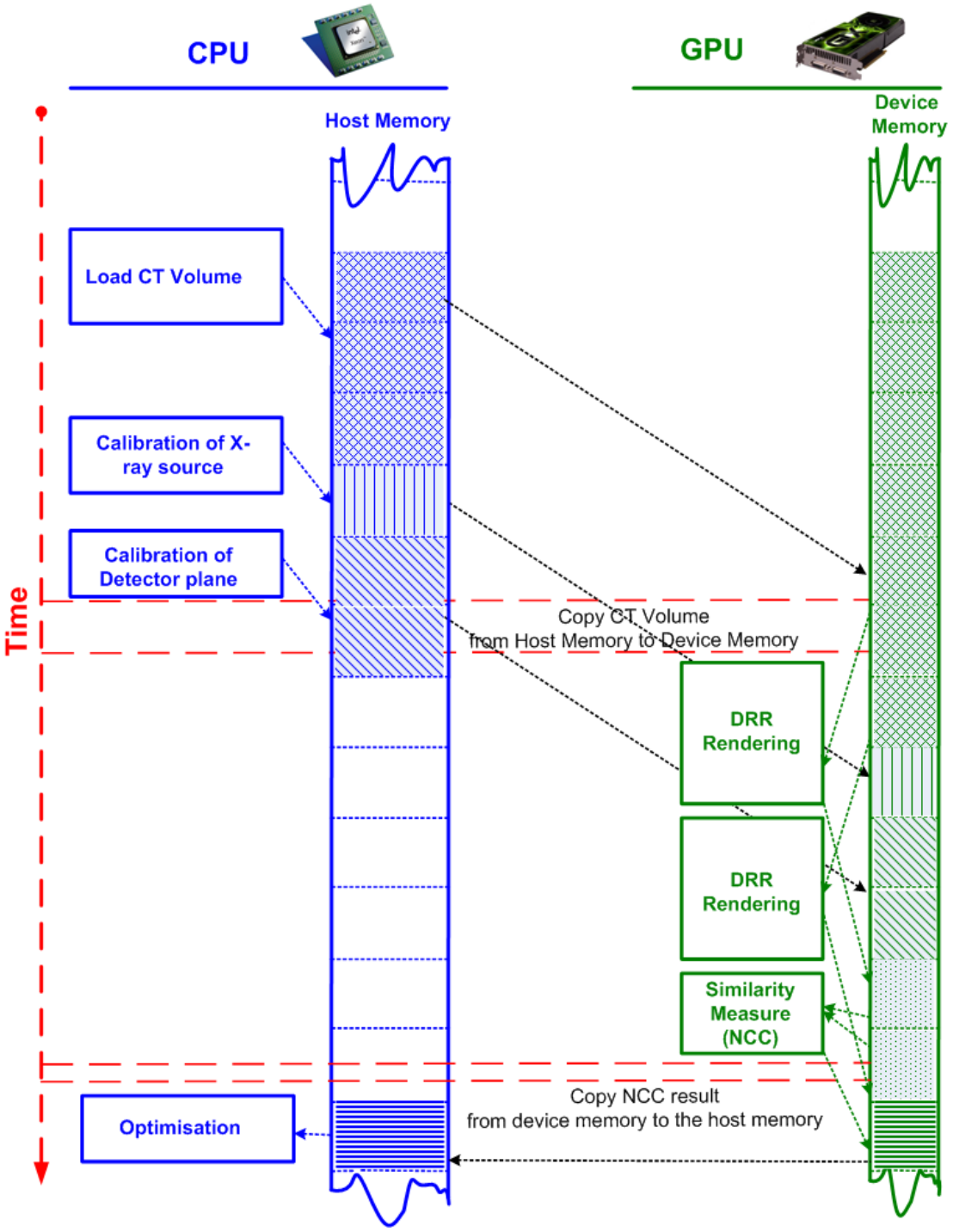


Figure 6.8: A single iteration of the registration process, where the DRR rendering implemented as a GPU-based rendering process and the similarity measure process is implemented as a GPU-based process.

Finally, the required memory space on the GPU card will vary according to the size of the CT volume and the rendered DRR image. The total required size in bytes
$\equiv 4(float) \times CT\_X\_dimension \times CT\_Y\_dimension \times CT\_Z\_dimension +$
$4(float) \times DetectorPlane\_X\_dimension \times DetectorPlane\_Y\_dimension +$
$4(float) \times DRR\_X\_dimension \times DRR\_Y\_dimension.$

For example the required device memory to render a DRR image of size 512×267 pixels from a CT volume of size $512 \times 512 \times 267$ voxels will be: $4 \times 512 \times 512 \times 267 + 4 \times 512 \times 267 + 4 \times 512 \times 267 \equiv 281063424$ bytes $\equiv 268$ MB.

6.6 Summary

In this chapter we accelerate the speed of the image registration process by enhancing the speed of rendering DRR images using the GPU. We proposed a hybrid CPU/GPU based registration solution that splits the registration process between the CPU and GPU to get the maximum performance (speed and accuracy). Using CUDA to implement a hardware solution for the DRR rendering process has improved the speed without compromising the DRR image quality. Comparing it to previous results using the CPU, however, the PSNR ratio was ∞ for all the different sizes of DRR images. Comparing our results of the DRR rendering process to previous work of other researchers we believe that we reported the fastest results rendering full resolution DRR images. We are able to render a DRR image from $256 \times 256 \times 133$ CT volume in about 24 ms using an NVidia GeForce 8800 GTX and in 2 ms usnig NVidia GeForce GTX 580. Finally, our results from this work can be built easily in to the system of 2D/3D registration using sparsely rendered DRR images, which will be illustrated in the next chapter.

Chapter 7

2D/3D Image Registration using Reduced Resolution and Sparsely Rendered DRR Images

In this chapter[1] we examine the performance of 2D/3D image registration using reduced resolution and sparsely rendered DRR images (i.e. rendering fragments of the DRR images). In Section 7.1 we describe the motivation of this work (approach). In Section 7.2, we investigate the relationship between the execution time of our parallel DRR algorithm, the number of cores, the number of rays (resolution), and the number of sampling points inside the CT volume which are used to render the DRR image. In Section 7.3, we discuss a local entropy measure (a statistical measure of the randomness), which we used to automatically select specific ROIs which are combined to form sparsely a rendered DRR. In Section 7.4, we integrate both of these techniques in a system for speeding up the registration process and evaluate it using a LINAC system providing both kV and MV reference images. We test its accuracy by comparing our results with those obtained using a commercial software package used at NNUH. In Section 7.5, we summarise the findings of this chapter.

[1] *This chapter is an adapted and extended version of: "O. Dorgham, M. Fisher, S.D. Laycock, A.J. Vinall and W. Holmes-Smith. Fast 2D/3D Image Registration using Accelerated Generation of Sparsely Rendered Digitally Reconstructed Radiographs.* International Journal of Computer Assisted Radiology and Surgery*, Volume 5, Supplement 1, pp. S68, June 2010".*

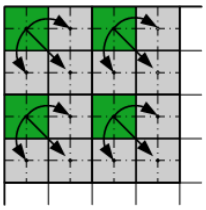
7.1 Motivation

This work is motivated by the need to further reduce the time within the 2D/3D registration loop. Here we focus on a number of approximate DRR rendering methods and investigate how these perform with respect to clinical registration accuracy.

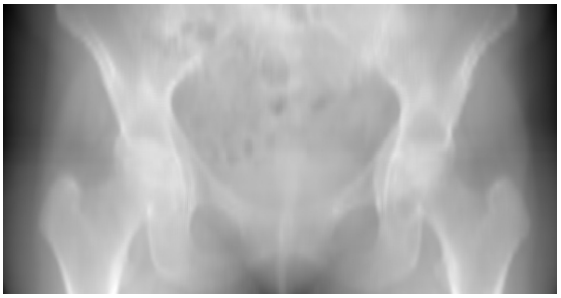
7.2 Rendering Reduced Resolution DRR Images

Rendering reduced resolution DRR images (RR-DRR), will consume less time in comparison to the full resolution DRR images (FR-DRR). We implement two methods of rendering RR-DRR images; firstly, by reducing the number of rays that are cast through the CT volume, and secondly, by reducing the number of sampling points inside the CT volume per each ray. Then we evaluate the quality and speed of rendered RR-DRR compared to that for FR-DRR images. In the first method we use nearest neighbour interpolation to replace all the missing values of the rendered DRR images [92]. We have the choice of controlling the number of rays intersecting the CT volume, so in our experiments we rendered RR-DRR images by using 25% of the rays that are used to render FR-DRR images. Hence, each ray intensity value is used to draw 4 pixels in the RR-DRR image, as shown in Figure 7.1.

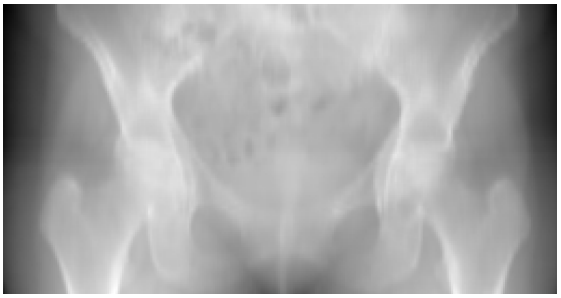
In the second method of rendering a RR-DRR image, we reduced the sampling points inside the CT volume to different ratios by 50% and 75% which provides a low cost computation to calculate the intensity value per each ray. A sample of the resulting RR-DRR images are presented in Figure 7.2. Speed of rendering RR-DRR images using single core is presented in Table 7.1 (however, speed of rendering RR-DRR images using different number of cores shows simillar results for rendring FR-DRR, values are given in Appendix A, Table A.11 and A.12, although, graphical representation is presented in Figure A.2).



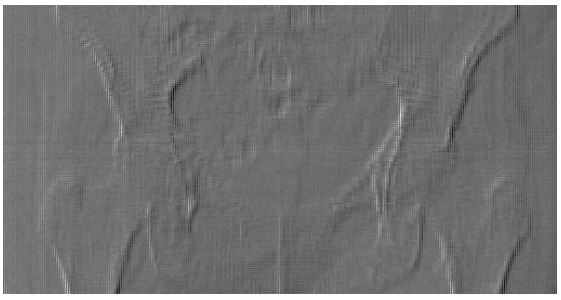
(a)



(b)

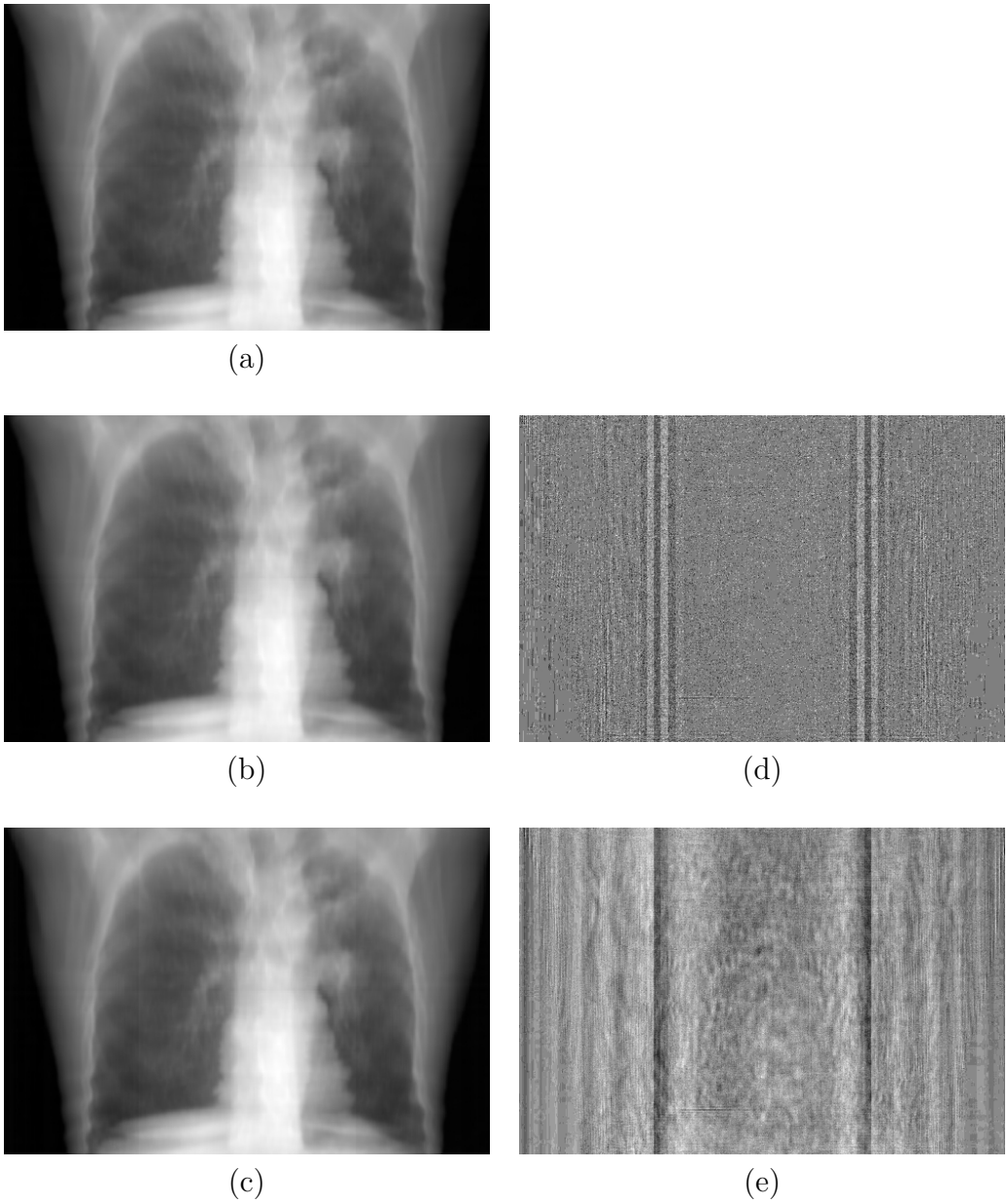


(c)



(d)

Figure 7.1: Example for RR-DRR using interpolated rays where (a) is a diagram shows a grid of nearest neighbour interpolation where green areas show the calculated pixels and the gray areas show the interpolated pixels in the DRR image, by calculating 25% of the required rays needed to render FR-DRR image, (b) FR-DRR and (c) RR-DRR image and (d) difference image formed by pixel by pixel subtraction between the two images (black pixel = no difference).



(a)

(b)

(d)

(c)

(e)

Figure 7.2: Example for RR-DRR using reduced sampling where(a) sample of Lung FR-DRR image, (b) and (c) RR-DRR images were rendered by reducing the sampling points by 50% and 75% respectively, (d) difference image formed by pixel by pixel subtraction between image (a) and image (b) while (e) difference image formed by pixel by pixel subtraction between image (a) and image (c) (black pixel = no difference).

	Speed of rendering RR-DRR images		
Method type / CT Volume Size	rays reduction	sampling method	
		50%	75%
$128 \times 128 \times 66$ (pelvis)	26 ms	53 ms	28 ms
$128 \times 128 \times 86$ (lung)	35 ms	72 ms	37 ms
$256 \times 256 \times 133$ (pelvis)	242 ms	429 ms	219 ms
$256 \times 256 \times 172$ (lung)	278 ms	573 ms	296 ms
$512 \times 512 \times 267$ (pelvis)	1586 ms	3153 ms	1691 ms
$512 \times 512 \times 344$(lung)	2008 ms	4105 ms	2201 ms

Table 7.1: Speed of rendering DRR images using the first and second method of rendering RR-DRR images using a single core of Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz.

By visual examination of the DRR images in Figure 7.1 and 7.2 it is hard to see the difference between the FR-DRR and RR-DRR images. But a quantitative comparison can be obtained by computing the PSNR (for more information about PSNR ratio, please refer to Section 6.4.1). A set of PSNR ratios were computed for a range of the most commonly used sizes (resolution) of DRR images and for two sets of CT volume (lung and pelvis), using the two methods (i.e. sampling and rays reduction) of rendering RR-DRR images discussed. PSNR values for FR-DRR images are also included (Table 7.2). Results from Table 7.2 illustrate why it is hard to notice the difference between our FR-DRR images and RR-DRR images in Figure 7.1 and 7.2. The PSNR ratio in Figure 7.1 is about 44 dB and in Figure 7.2 is about 43 dB. According to Huang et al.[151], PSNR ratios above 36 dB represent an excellent image quality of compressed images (in our case RR-DRR images). Although, results form Table 7.2 show that using 25% of the total number of rays (first method of rendering RR-DRR) is at the breaking point (i.e. where the registration fail) of rendering RR-DRR from CT volume of size $256 \times 256 \times 133$ voxels (pelvis) and $256 \times 256 \times 172$ voxels (lung), also reducing the number of sample points inside the CT volume to 75% is the breaking point of rendering RR-DRR images from CT volume $256 \times 256 \times 172$ voxels (lung). Comparing the method of rendering RR-DRR images to the method of multi-

scale pyramid, we render DRR images with the consistent resolution (i.e. full or reduced) in all the registration iterations. On the other hand, multi-scale pyramid method could give better registration results but this could cost more time to render different DRR images with different resolutions. However, we render a RR-DRR image from CT volume of size $256 \times 256 \times 133$ voxels using the first method of reducing the number of calculated rays with a resolution of 256×133 pixels in approximately 242 ms, and we render a RR-DRR image from the same CT volume using the second method of reducing the sampling points for 25% in the CT volume, with a resolution of 256×133 pixels in approximately 219 ms using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz.

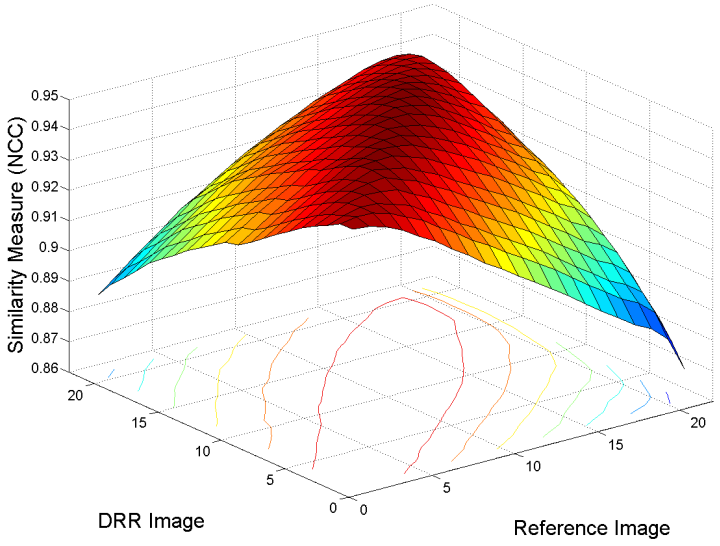
	PSNR ratio		
Method type / CT Volume Size	rays reduction	sampling method	
		50%	75%
$128 \times 128 \times 66$ (pelvis)	32.62 dB	33.49 dB	29.21 dB
$128 \times 128 \times 86$ (lung)	30.64 dB	35.65 dB	30.35 dB
$256 \times 256 \times 133$ (pelvis)	35.87 dB	44.18 dB	33.75 dB
$256 \times 256 \times 172$ (lung)	36.49 dB	46.29 dB	35.58 dB
$512 \times 512 \times 267$ (pelvis)	43.72 dB	53.23 dB	43.66 dB
$512 \times 512 \times 344$(lung)	42.49 dB	52.87 dB	45.05dB

Table 7.2: A set of PSNR ratios between DRR images rendered using the first and second method of rendering RR-DRR images and FR-DRR images.

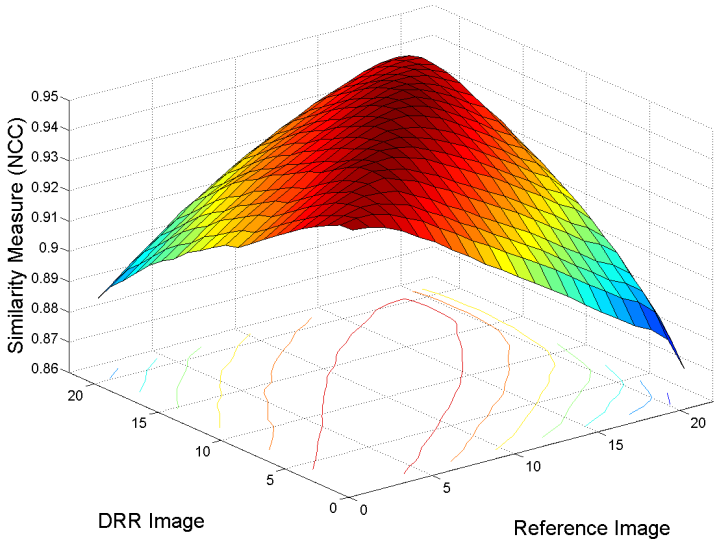
Referring to our methods of rendering reduced resolution DRR images. We compared the results of performing the 2D/3D registration process using FR-DRR images and RR-DRR images. Figures 7.3 and 7.4 show the NCC surface produced when matching reference and floating DRR images (at full and reduced resolutions) in the range $0° \rightarrow +20°$ (the values are given in Appendix A, Tables A.2 - A.9). Figure 7.5 indicates the floating DRR image which is best match in each case (values tabulated in A.10). These confirm that FR-DRRs and RR-DRR's (50% under-sampled) produce the most accurate results, most cases being able to give perfect registration performance. Registration at large angles (i.e. $> 17°$) tend to

produce large errors because our DRR algorithm moved on ideal geometry (i.e. a point X-ray source) and the Linac source is not ideal. Hence there are errors between DRR and epid images but these are much significant at large angles. This can be seen in Figure 7.15.

However, the performance of the registration process using RR-DRR with 75% did not provide accurate results, either using the sampling method nor the rays reducing method. This registration results explained by the previous PSNR ratios that have been presented in Table 7.2 as the PSNR values where less than 36 dB for the 256×133 pixels of pelvis RR-DRR image.

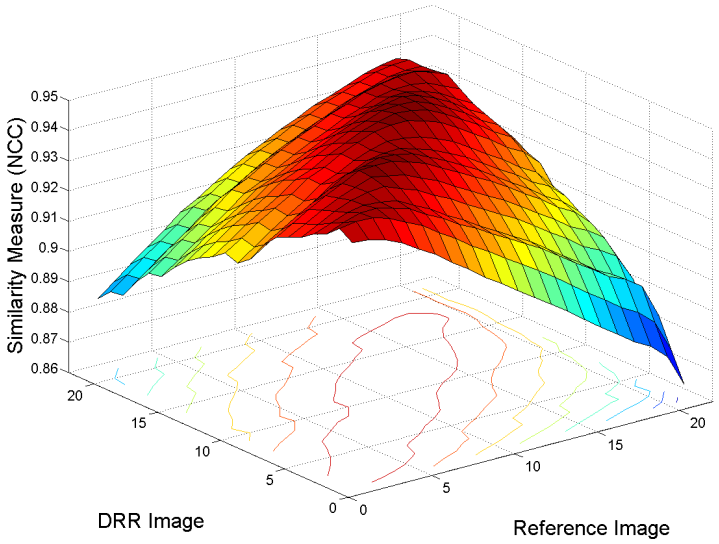


(a)

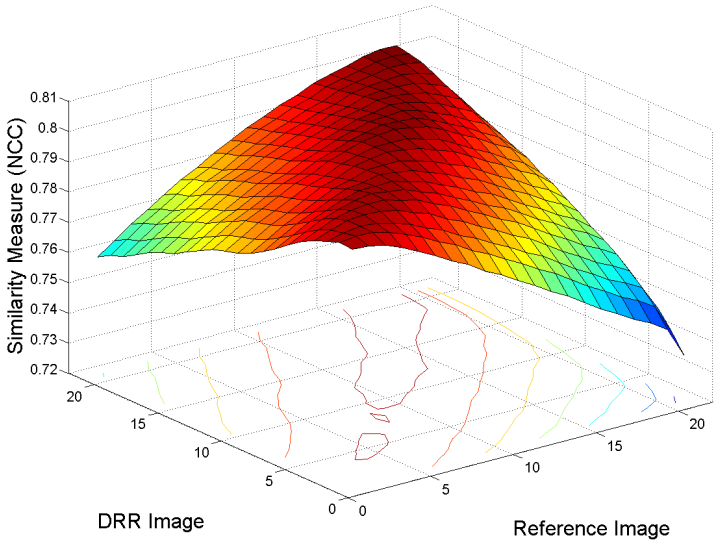


(b)

Figure 7.3: Results of performing 2D/3D registration process between kV reference image and the DRR images in range of $0° \rightarrow +20°$. Where (a) results of using FR-DRR images and (b) results of using RR-DRR images for sampling method with 50% samples of the volume.



(c)



(d)

Figure 7.4: Results of performing 2D/3D registration process between kV reference image and the DRR images in range of $0° \rightarrow +20°$. Where (c) results of using RR-DRR images for sampling method with 75% reduction in samples, and (d) results of using RR-DRR images for the method of reducing number of rays with 75%.

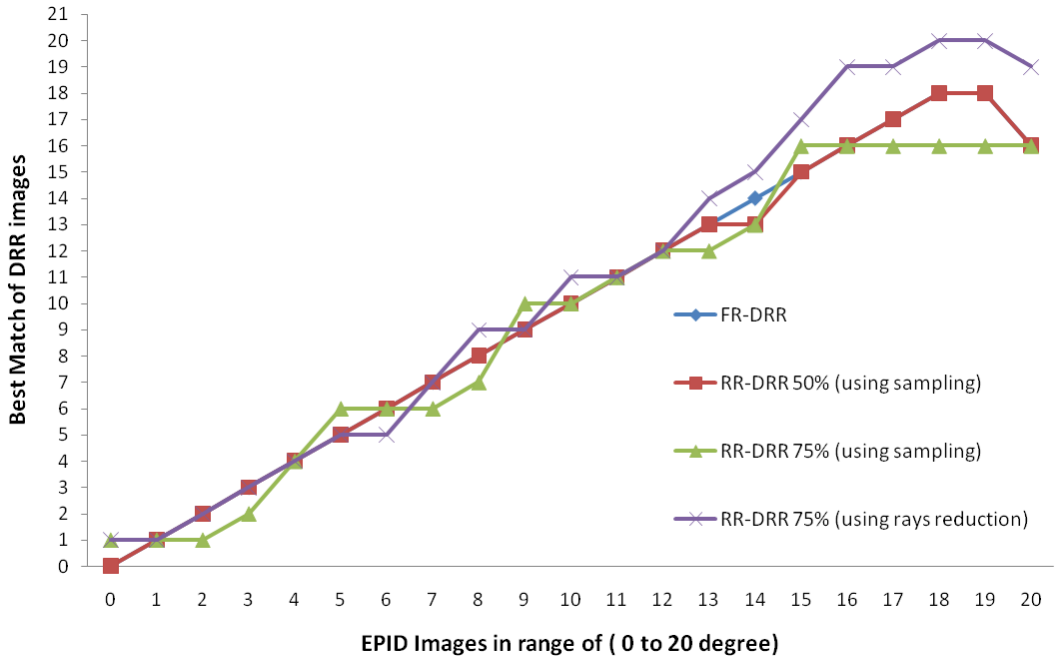


Figure 7.5: Accuracy for the performance of the 2D/3D registration using different methods of rendering the DRR images (full and reduced resolution images).

7.3 Sparsely Rendered DRR Images

We illustrated in the previous section that the total time of rendering DRR images can be reduced by reducing the resolution of the DRR images either by casting fewer rays or by taking fewer samples inside the CT volume. In this section we will discuss a further approach for speeding up the rendering of DRR images (for registration) by rendering only part of the image. To reduce the time of rendering full size DRR images, we used features within the reference images to select a region of the DRR image to be rendered and subsequently used in the registration process. We used an entropy measure to select these region of interest (ROI) areas in the DRR images, as we illustrate in the following sections.

7.3.1 Local Entropy Measure

In 1948 Shannon introduced a universal measure (Entropy) to model and quantify information using probability theory [150]. Entropy is a statistical measure of randomness that can be used to indicate the amount of information in an image [50][58], it is used in image processing in different applications like image retrieval, classification or image segmentation [155][6]. We used the entropy measure to weight different features within both kV or MV energy reference images (e.g, bone, muscle tissue, air, etc.). We calculate entropy as follows:

$$E = -\sum_{i=0}^{255}(p(i) \times log\ p(i))$$

Where $p(i)$ is the probability of the i th intensity value of the histogram ($i = 0 \rightarrow 255$) for the local regions of our gray scale images $p(i) = Histogram\,[i] \div sizeofImage$. Entropy is 0 for a stable image (an image with pixels having the same value), as values are in the range $[0, 1]$ and calculating E with values of $p(i) = 0$ and $p(i) = 1$ will results of $E = 0$. Entropy will be maximum when all values are equal [50]. Using maximum local entropy values we are able to identify regions with significant features, for example, Figure 7.6 illustrates maximum entropy regions in a coronal Lung CT slice (with various region sizes). In the 2D/3D registration the entropy measure can be used to select between different ROI areas. More details about using the entropy measure in the 2D/3D registration are described in the following section.

7.3.2 Automatic Selection of Region of Interest

As we described in Chapter 3, Section 3.4, rendering DRR images conventionally demands a massive number of calculations to find the intersection points between X-rays and voxels within the CT volume (i.e. large number of ray casting operations). To reduce the computational load incurred by repeatedly rendering full size DRR images (FS-DRR) we render DRR image fragments for specific regions of interest. ROI within the reference image are chosen automatically then the mask is projected to render fragments the of the DRR image. The fragment size is predetermined and the entropy value can be used to denote the information

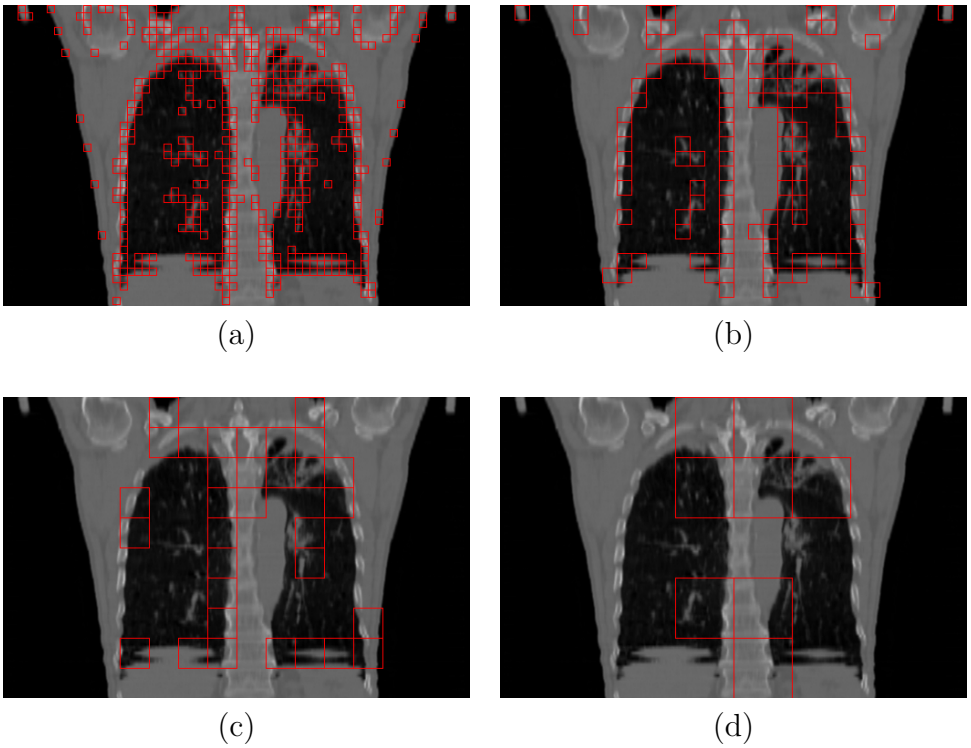
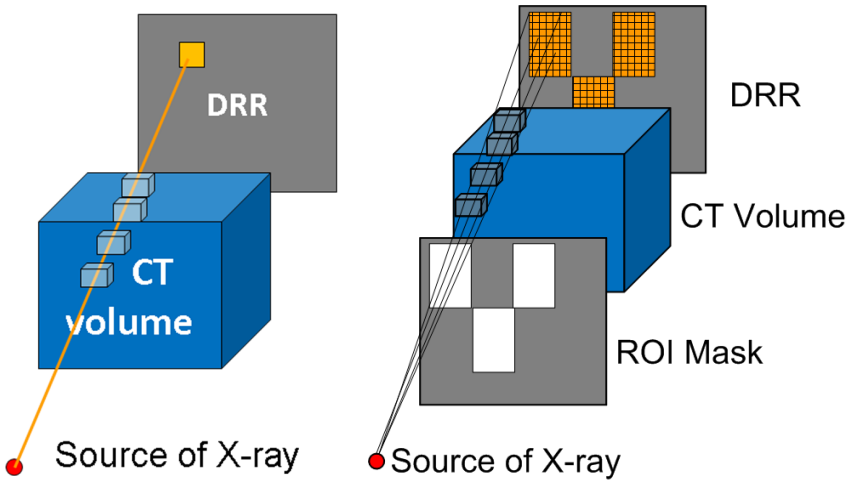


Figure 7.6: ROI selected using entropy measure with different blocks sizes of selected areas of 20%. for a coronal Lung CT (512×330 pixels) slice image entropy measured for blocks of size (a) 8×8 pixels, (b) 16×16 pixels, (c) 32×32 pixels and (d) 64×64 pixels.

content of the ROI [191]. We sort the region fragments in descending order according to their entropy values; regions with highest entropy are selected as a ROI fragments, which will be used as a mask for DRR rendering step as shown in Figure 7.7. Instead of rendering a FS-DRR, we render selected spaces of the volume determined by the ROI mask areas. Sparse DRR images significantly reduce in the number of rays cast and hence time. Examples of sparse DRR images are shown in Figure 7.8.

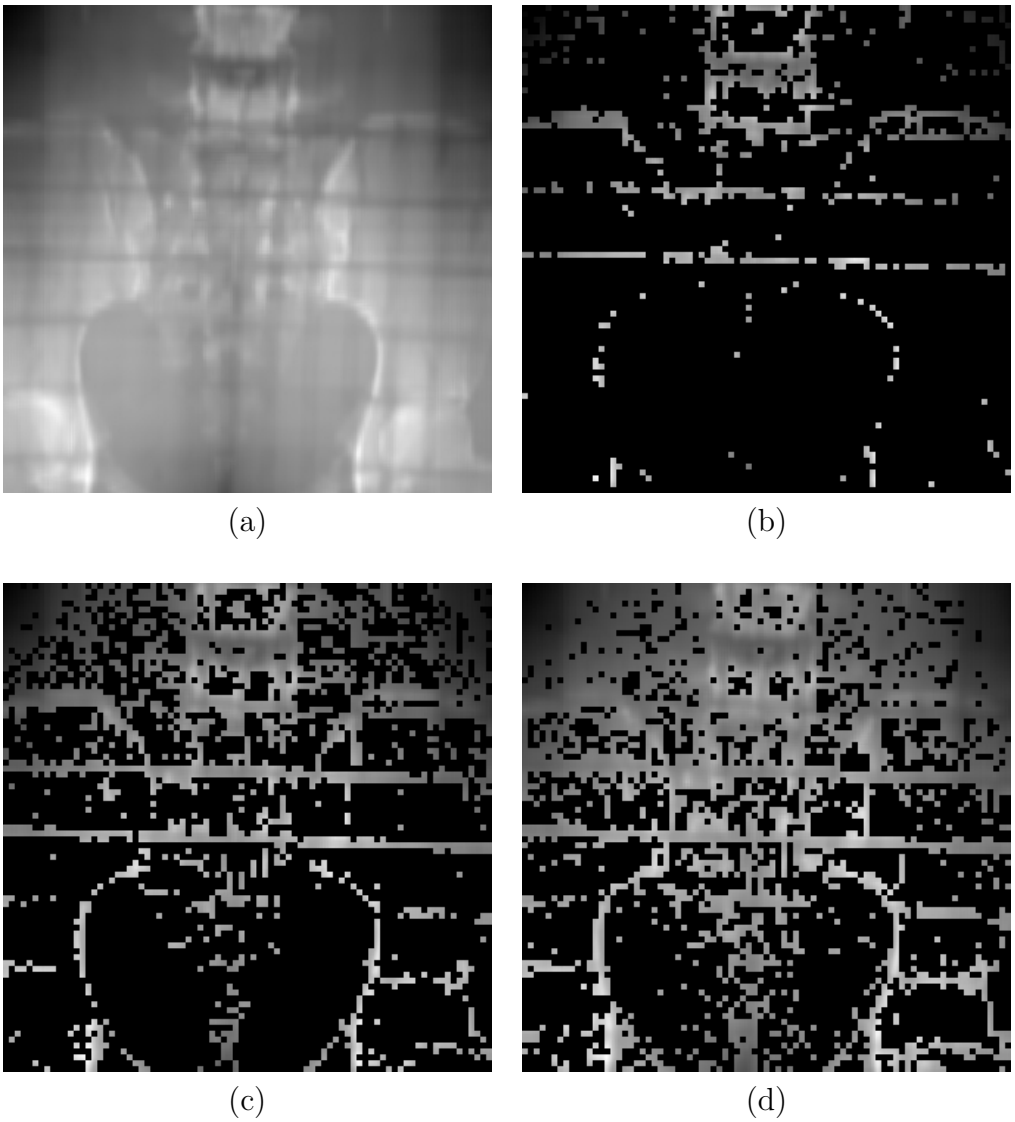
There are two parameters related to the ROI fragments, firstly, the size of fragment which is measured by $(n \times m)$ pixels as in Figure 7.6, secondly, the number of fragments which is measured by the percentage of FS-DRR coverage as in Figure 7.8. The size of the ROI area and its blocks can affect the accuracy and speed of the registration process, more details and experimental results are



(a) (b)

Figure 7.7: (a) Conventional method of DRR rendering, (b) sparse DRR rendering.

discussed and presented in the following section. By selecting ROI automatically, we are eliminating human error and providing a stable, robust and fast method of selecting the areas of ROI, thereby enabling registration process to be performed in an accurate way without any concern about the experience of the person selecting the ROI used in the registration process. However, the execution time of rendering sparsely DRR images differ according to the size of the ROI area and the number of the cores are used to render the sparse DRR image. The size of the ROI determines the required number of rays needed in the registration process, and in turn the number of used cores and the number of parallel threads which are used to cast multiple rays simultaneously when parallel architectures are used. This, in turn reduces the time of rendering sparsely DRR images (SP-DRR), for example rendering SP-DRR image 256×133 pixels, for example, 30% of ROI requires about 60 ms for FR-DRR image and about 15 ms for RR-DRR image rendered from CT volume of size $256 \times 256 \times 133$ pixels using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz. Possible improvement in speed might be obtained by selecting sets of ROI randomly. However, this strategy could fail to return useful ROI and so we feel that selecting the most useful ROIs derives more robust registration results.



(a) (b)

(c) (d)

Figure 7.8: Example of DRR images where (a) original FS-DRR image, (b-d) represent sparse DRRs formed using 10%, 30% and 50% of ROI fragments respectively.

7.4 Using Fast DRR Rendering in Image Registration with a LINAC Radiation Therapy System

Many different types of radiation therapy systems are used in cancer treatment, such as: Linear Accelerator (LINAC), CyberKnife and Tomotherapy. As we described in earlier stage in Section 2.4, and despite the difference in the structure of the systems but the objective is similar [41] and patient setup is needed in all cases where treatment is delivered in fractions. For patient setup using any of the previous systems, we need to render accurate and quick DRR images consistent with the geometric configuration of the Electronic Portal Imaging Device (EPID) which used to produce reference images from the desired system.

7.4.1 LINAC System

In our study, we used a Varian 2100EX LINAC radiation therapy system fitted with Varian AS1000 EPID panel (for MV operation) and IDU20 EPID panel (for kV operation), located at the Norwich Norfolk University Hospital (NNUH). The geometry is as shown in Figure 7.9.

The LINAC was set up using a field size of 20×20 cm defined at the iso-centre i.e. 100 cm from the X-ray source (we assume X-rays emanate from a point source and strike a flat panel situated behind the patient, i.e. conventional 'C' arm geometry). The EPID panel is located 40 cm beyond the iso-centre. The detector area is 40.14×30.11 cm with 512×384 pixels and 0.784 pixel pitch (Note: the active area of the detector plane is 30×24 cm for this geometry but will vary slightly if the detector plane is moved). The couch or patient support system (PSS) can be rotated and translated in six degrees of freedom (DoF), however to simplify our experimental simulation we only attempt to recover rotation around the z-axis (i.e. 1 DoF). Depending on the type of radiation therapy treatment system, 2D/3D image registration could be a kiloVoltage/kiloVoltage (kV/kV) registration process (e.g. Cyberknife) or, it could be a MegaVoltage/kiloVoltage (MV/kV) registration process (e.g. most conventional LINAC systems). However,

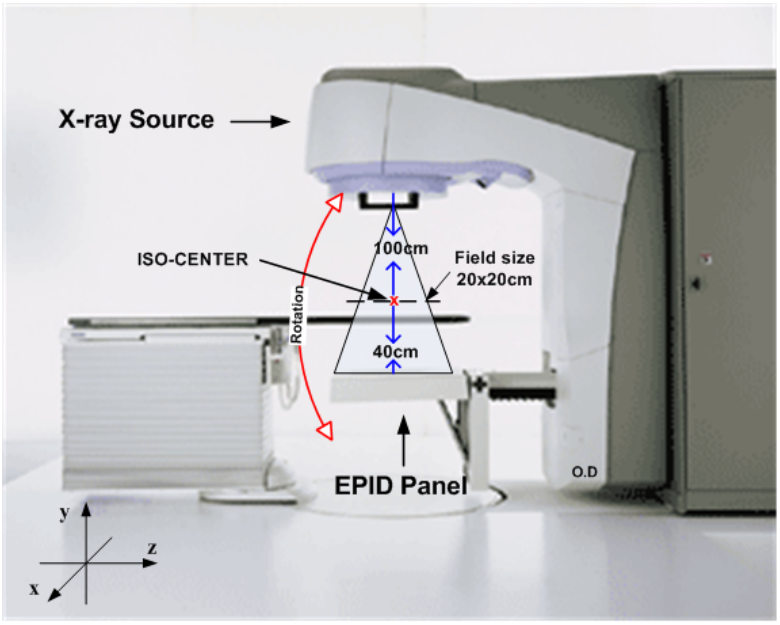


Figure 7.9: Details of the radiation therapy system used.

as the Varian 2100EX LINAC system we used for this work able to render both kV and MV reference images (see Figure 7.10), we can investigate both techniques.

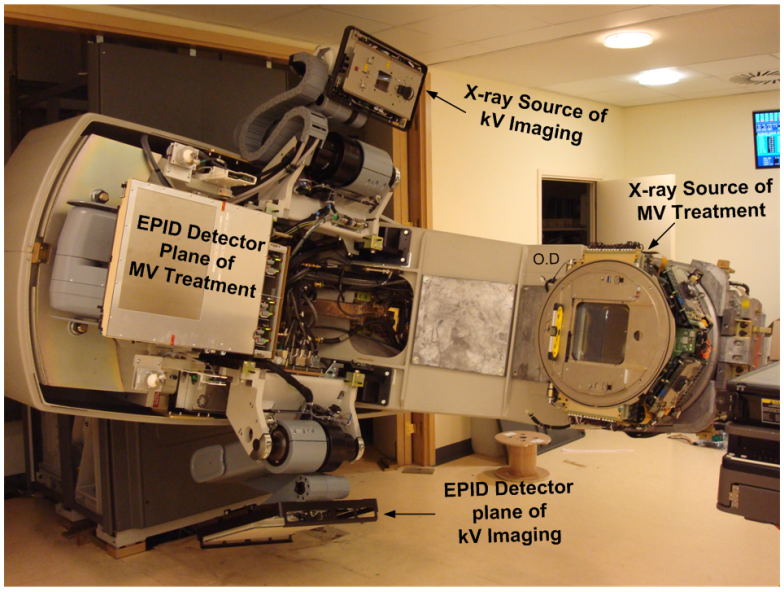
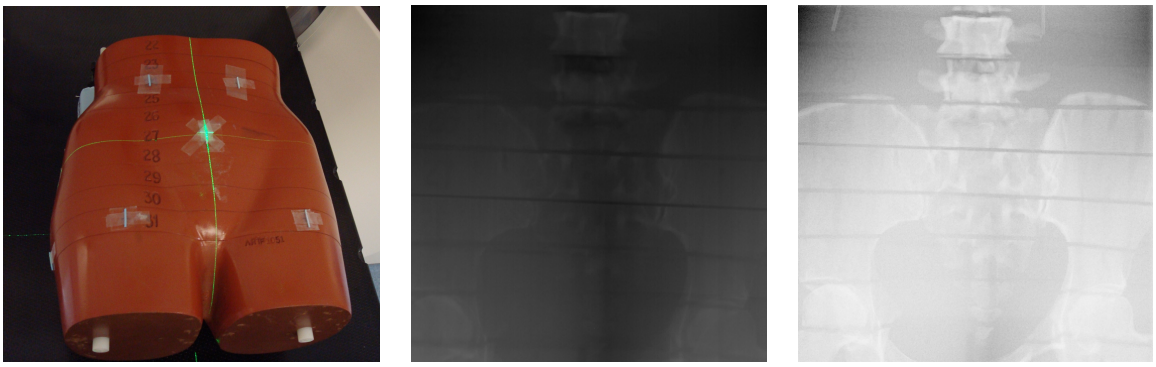


Figure 7.10: Varian 2100EX LINAC system in NNUH, which is able to render dual type of reference images (MV and kV images).

We used a Varian 2100EX set to produce X-rays in the $(79 - 81)$ kV energy range, where the kV images were taken using a function called Automatic Exposure Control (AEC) so the exposure levels are automatically set. The energy selected does not vary very much. Most lateral images (-20 and 20 degree) are captured at 81 kV and the most anterior images (0 degree) 79 kV. For MV experiments we used the Varian 2100EX at 6 MV energy. A Rando phantom was used to provide CT volume and to render the kV and MV energy reference images. The experimental system is shown in Figure 7.11.



(a) (b) (c)

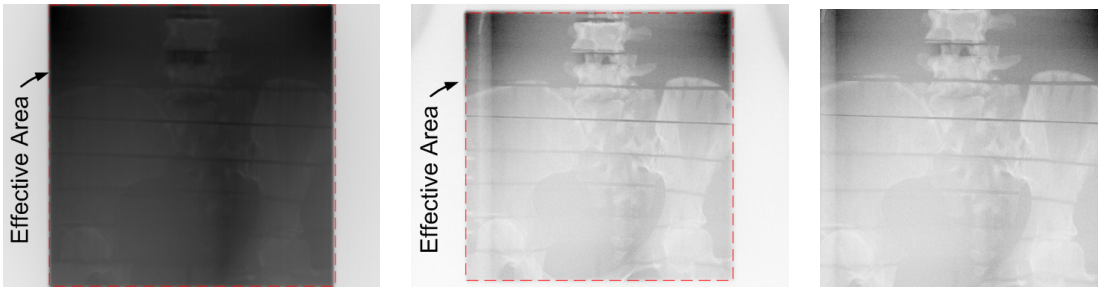
Figure 7.11: (a)Pelvis Rando phantom with EPID images of (b) MV energy of X-rays, and (c) KV energy of X-rays.

We build our CT volume from a series of Dicom images obtained from the Rando phantom and collected as axial 3 mm CT slices with a pixel spacing of 0.96289×0.96289 mm using 120 kV energy X-rays using GE HiSpeed FX/i CT scanner located at NNUH.

7.4.2 Performance of (kV/kV) and (MV/kV) Radiation Therapy Systems

Using the geometry specification of the LINAC system we rendered DRRs exactly equal in size to the MV and kV reference images in 1° steps within a range of $-20° \rightarrow +20°$ in R_z. However, after we got the images from the hospital we found there is a variation in the size of the effective area between the MV and kV images as illustrated in Figure 7.12(a,b). We believe this variation was due to a

difference in the location of EPID panel geometry between both types of reference images. To compensate the variations in geometry (between MV and kV setup) we perform 2D/3D registration on images of 332×332 pixels by cropping the both kV and MV reference images as illustrated in Figure 7.12(c).



(a) (b) (c)

Figure 7.12: (a,b) Original size of EPID images of MV and kV energy of X-rays and (c) cropped kV EPID image (Note: the Rando phantom artefact).

We render DRR images of the same size as the cropped reference images of 332×332 pixels prior to the registration process between the DRR images and each of the reference images without bothering about the difference in the effective area sizes. As we do not know the exact location of the iso-centre (so called blind iso-centre) we use the 2D/3D registration process to recover it. We use the curved lines (an artefact produced by the Rando phantom slices) to confirm we have the correct geometry (i.e. the X-ray source is in front of the panel and not vice-versa). Note: the curve artefacts are also shifted due to a small misalignment in the geometry (discovered after the data set was acquired and verified by commercial Varian's offline review software).

We present results confirming the sensitivity and accuracy of our method. First we initialise the system at predefined iso-centre. To illustrate the sensitivity of the similarity measure we choose an EPID image in the range $0° \rightarrow +20°$ ($0° \rightarrow +20°$ or $-20° \rightarrow 0°$ shows the same results). Next we measure the similarity between the reference and each of 20 DRR images (full size or sparse ones) from $0° \rightarrow +20°$ rotation around the z axis, and present these results in Table 7.3 (reference image at 10°). Figure 7.13 shows the EPID images and the best match of the DRR images, which confirming that the system can reliably recover rotations over a

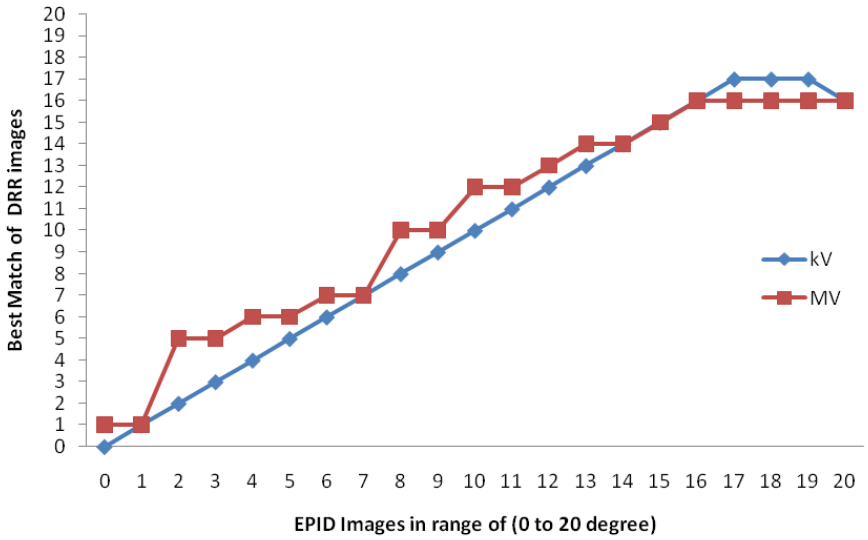


Figure 7.13: Accuracy of recovered rotation (using full size DRRs).

limited angle of view (up to 17° around the z axis).

Unfortunately the results are not so good for the MV EPID images, almost certainly due to their poor quality (compared to the kV ones). Figure 7.14 shows an example of the results that we obtain by performing the best-matching operation for the 10° EPID image and sparsely rendered DRR images with different percentages of automatic selected ROI. These confirm that for kV/kV registrations (10% of ROI) is sufficient for reliable 2D/3D registration. The performance of an MV/kV system is worse, although the error performance might be improved by applying some image enhancement. Using 10% of the sparse DRR image as a ROI will increase the speed of sparse DRR rendering and the similarity measure algorithm as the NCC value is calculated only for the selected ROI. To assess the absolute error in the registration we used Varian's offline review software to register our rendered DRR images (floating images) from a pre-collected CT volume for pelvis random phantom with reference kV images generated using the same rando phantom. Registration results shows difference between floating DRR images and reference kV images which shown in Figure 7.15.

DRR	0°	1°	2°	3°
NCC	0.9144	0.9207	0.9258	0.9287
DRR	4°	5°	6°	7°
NCC	0.9332	0.93324	0.941	0.9437
DRR	8°	9°	**10°**	11°
NCC	0.9459	0.94696	**0.94699**	0.9458
DRR	12°	13°	14°	15°
NCC	0.944	0.9421	0.9398	0.9374
DRR	16°	17°	18°	19°
NCC	0.9344	0.9314	0.9279	0.9246

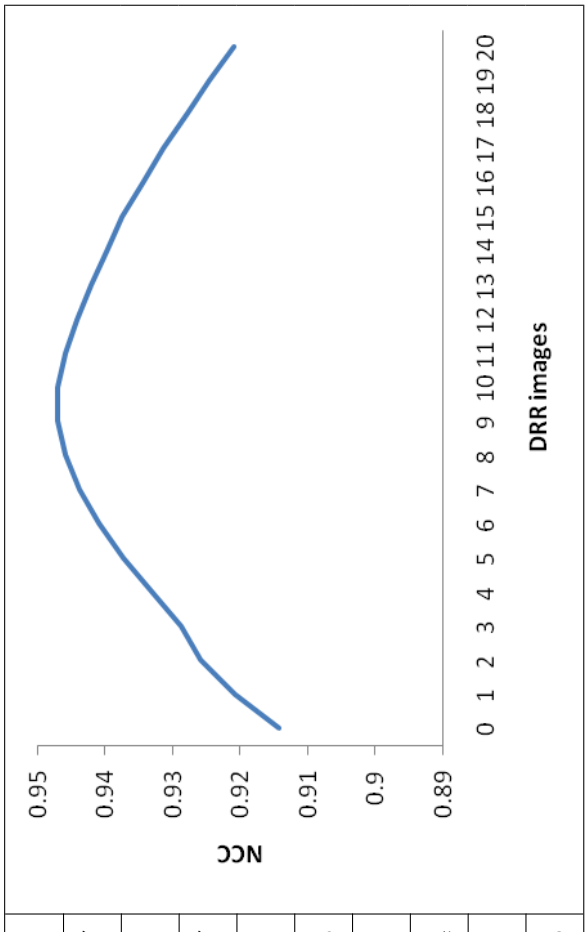


Table 7.3: Sensitivity of NCC similarity measure.

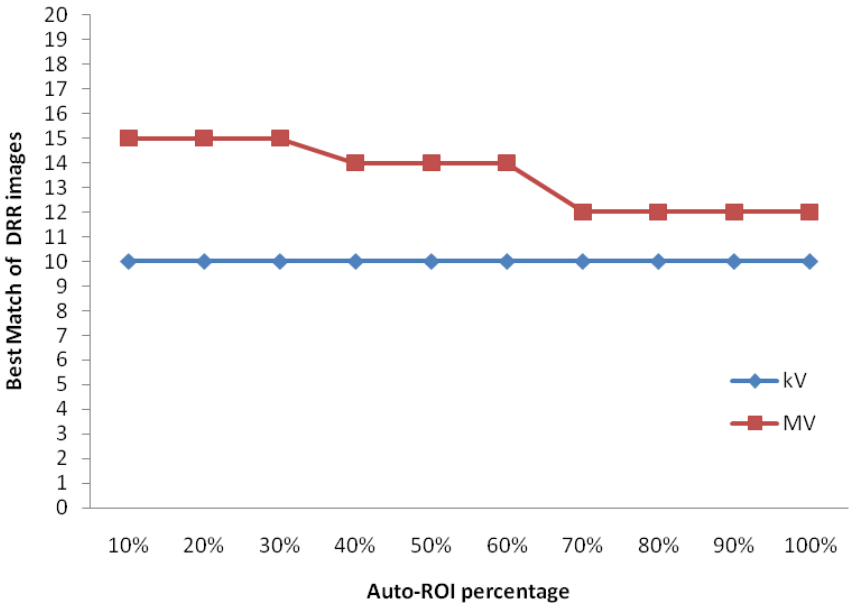


Figure 7.14: Results of 10 iterations of performing best matching operation between 10° EPID image (kV, MV) and different ratios of sparse DRR images in range of $0° \rightarrow +20°$.

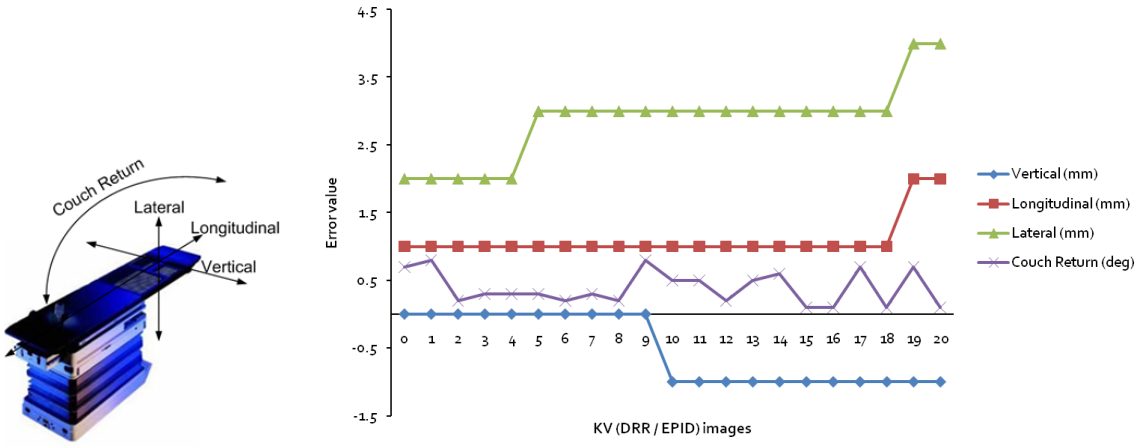


Figure 7.15: Directional error of kV (DRR/EPID) images using NNUH Varian commercial software.

Overall, we can say that these results are in the range of clinical radiotherapy treatment errors (according to [60]) giving target registration errors of ≤ 3 mm in lateral direction, ≤ 1 mm in longitudinal direction, ≤ 1 mm in vertical direction and $< 1^\circ$ in couch return direction for DRR and EPID images that are rendered in the range of $0^\circ \rightarrow +18^\circ$ rotational movement around z axis. However, larger registration error values could be resulted from this experiment as we were using Varian's commercial software which could include another errors that are in the range of clinical radiotherapy treatment (which we are not sure from it as it is commercial software). Despite this issue we were able to perform the registration process using our developed method with high accuracy as we illustrated earlier in Figure 7.13.

7.5 Summary

In this chapter, we developed a method of accelerating the process of 2D/3D image registration by rendering reduced resolution and sparse DRR images. This was motivated by a need to improve the speed of this process while maintaining sufficient clinical accuracy. The speed of rendering DRR images depends mainly on the resolution of DRR images and the number and size of the ROI blocks selected for sparse rendering using the entropy measure. Small numbers of ROI means a faster rendering process for DRR images; consequently faster 2D/3D image registration. We tested and evaluated the performance of the accelerated 2D/3D registration process with both kV and MV reference images using data from an upgraded Varian LINAC machine at the NNUH. The CT data and EPID images were derived from a Rando phantom. The experimental procedure recovered a rotation in the direction of R_z in a range of $-20^\circ \rightarrow +20^\circ$ (in 1° steps). We estimated absolute target registration errors by comparing our registrations to those recovered using a commercial Varian's offline review software.

Experimental results of registering kV EPID images to floating sparse DRR (kV) images shows that it is possible to perform the registration process with a level of accuracy sufficient for most pelvic cancer sites for rotations up to 17° around z axis, but results for the MV EPID images were not so good because of their low quality (compared to the kV ones). Moreover, the kV/kV results remain

acceptable even when rendering sparse DRR images at 10% of ROI. Overall, using kV/kV registration it is possible to recover a movement of $\leq$ 3 mm in lateral direction , $\leq$ 1 mm in longitudinal direction, $\leq$ 1 mm in vertical direction and $<$ 1° in couch return direction for DRR and EPID images rendered in range $0° \rightarrow +18°$ rotational movement around z axis. These results have been collected using single dataset for pelvis rando phantom, but we believe results will not be affected that much using different datasets (e.g. lung or head) as all of these body sites does share bone and soft tissues. Furthermore, the quality of DRR images is still in the excellent resolution category with PSNR values of about 36 dB for lung RR-DRR image with 256×172 pixels resolution rendered from a $256 \times 256 \times 172$ CT volume using the two methods of rendering RR-DRR images. For example, reducing the number of rays (i.e. 25% of the total number of rays) or reducing the number of sample points (i.e. 75% reduction on the total number of sample points). Results show that the performance of the 2D/3D registration is almost the same using the FR-DRR images and RR-DRR images with 50% using the sampling method. However, the performance of the registration process using pelvis RR-DRR with 75% sampling did not provide an accurate result, either using the sampling method or by reducing the rays cast. This registration results are explained by the PSNR ratios which were less than 36 dB for the 256×133 pixels pelvis RR-DRR image. Finally, combining parallel processing with sparse DRR rendering using the automatic method of ROI selection increases the speed of the registration process according to the size of the ROI. Acceptable registration performance even can be achieved with only a few ROIs.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis described a number of methods aimed at accelerating 2D/3D image registration which have been developed by the author and other researchers. We developed our methods using a strategy that enables us to maximise the benefit of available software and hardware resources (GPU, if available; otherwise multiple CPU cores, or single core), keeping in mind that the speed we gained should not interfere with the clinical accuracy of 2D/3D image registration. However, we conclude this work by answering the following questions.

What strategy did we adopt to accelerate 2D/3D image registration?

We studied 2D/3D image registration as a collection of three main processes: the rendering of digitally reconstructed radiographs, similarity measurement and optimisation processes. Our development was guided based on the computational complexity (time and space) of the underlying algorithms. Accelerating the DRR rendering process was the highest priority as it is the most complex process and forms an acknowledged bottleneck in 2D/3D image registration. In later steps we built the other 2D/3D registration components in such a way to make them to be compatible to each other.

What type of methods did we develop to accelerate DRR rendering?

Based on the nature of the DRR rendering process we developed a number of different strategies to accelerate it. Reducing the number of intersection points between X-rays and CT volume was one of the solutions which we adopted, one way of accomplishing this was by compressing the CT volume, other solutions we tested exploited sub-sampling both inside the CT volume and by reducing the number of rays that intersect the CT volume. Moreover, we combined the previous methods in parallel processing approaches that resulted in both software (CPU) and hardware (GPU) solutions. Through automatic rendering of sparse DRR images we also demonstrated an important approach for accelerating rendering process as we are rendering only a fraction of the image. Finally, all methods (i.e. paralleisation, reduced resolution and automatic selection of ROI) are combined in one registration system as shown in Figure A.1 .

What speedup was gained by accelerating the DRR rendering algorithm?

The acceleration we achieved varies according to the methods we implemented to render DRRs and the quality of images we require. Using the Octree method of compressing voxelised CT volumes it takes 390 ms to render a DRR image of size 512×384 pixels from a $512 \times 512 \times 384$ voxel compressed CT volumetric data set using an Intel® Core™ 2 Duo Processor T7200 with 4M Cache running at 2.00 GHz. In another method we used parallel programming to accelerate the DRR rendering process. This takes $\sim$ 200 ms to render a full resolution DRR image of size 256×133 pixels from CT volume of size $256 \times 256 \times 133$ voxels using a Dell Precision Workstation with Intel® T5500 Dual Quad core processor running at 2.3GHz. As expected, reducing the DRR image resolution improved the speed of DRR rendering. It takes $\sim$ 50 ms to render a DRR image of size 256×133 pixels by reducing the number of rays cast by 25% compared with that used to render full resolution DRR images. Reducing the resolution of DRR images by taking fewer samples inside the CT volume was also investigated, this takes $\sim$ 100 ms to render a DRR image of size 256×133 pixels from a CT volume of size

$256 \times 256 \times 133$ voxels with 50% of the full resolution DRR image and using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz. The most successful method we developed is a hardware solution which is able to **render a DRR image from a** $256 \times 256 \times 133$ **CT volume in** ~ 24 **ms** using an NVidia GeForce 8800 GTX and in ~ 2 **ms** using NVidia GeForce GTX 580. According to our knowledge this result is one of the fastest DRR rendering results to be achieved to date.

How do we measure the quality of the accelerated DRR images that we rendered?

We measure the quality of the accelerated DRR images by performing a quantitative comparison by computing peak signal-to-noise ratio (PSNR) of images rendered at full resolution using a reference algorithm and those rendered using an accelerated approach.

The two methods of rendering RR-DRR images we tested (i.e. reducing the number of rays by 75% or the number of sample points by 50% and 75%), both returned DRR images that were classified as excellent with PSNR values of about 36 dB for RR-DRR lung images with 256×172 pixels resolution rendered from a $256 \times 256 \times 172$ CT volume. Additionally, using these algorithms within a 2D/3D registration framework showed that the performance is almost the same using full resolution DRR images and reduced resolution DRR images with 50% down sampling. However, the performance of the registration process using pelvis reduced resolution DRR with 75% down sampling did not provide accurate results, neither using the reduced sampling method or the ray reducing method. This registration result can be explained by the PSNR ratios as this was less than 36 dB for the 256×133 pixels of pelvis reduced resolution DRR image. Using CUDA to implement a GPU hardware solution for the DRR rendering process improved the speed without compromising the DRR image quality. When compared with reference DRRs all those produced by the GPU exhibited a PSNR ratio of ∞ (even though GPU arithmetic is single precision).

What effect does accelerating 2D/3D registration have on clinical accuracy?

Methods of accelerating registration using Octree compression can still achieve results with in acceptable accuracy limits of clinical accuracy. Using a Octree compressed CT volume for 2D/3D image registration can recover patient support system (PSS) {x,y,z} translation of up to ±10 mm with sub-voxel accuracy (i.e $\ll$ 2 mm) and angular {pitch, yaw, roll} rotations up to ±10 degrees with an accuracy of better than 1 degree. Furthermore, the performance does not degrade significantly with high levels of compression (>80%). The errors in recovered y translations are larger than in x and z since the X-ray fan beam is nearly parallel ($\sim 4.5^\circ$) and so the geometry is insensitive to adjustments in the height of the couch. In practice, the height of the couch (i.e. patient) much less likely to change compared to x, y translation and p, w, r rotation of the anatomy and so this is not seen as a significant problem.

Using the method of accelerating the process of 2D/3D image registration by parallelising and sparsely rendering the DRR images also achieved acceptable results. We tested and evaluated the performance of the accelerated 2D/3D registration process with both kV and MV reference images using data from an upgraded Varian LINAC machine at the NNUH. The CT data and EPID images were derived from a Rando phantom. The experimental procedure recovered a rotation the direction of R_z in a range of $-20^\circ \rightarrow +20^\circ$ (in 1° steps). We estimated absolute target registration errors by comparing our registrations to those recovered using a commercial Varian's offline review software. Experimental results of registering kV EPID images to floating sparse DRR (kV) images shows that it is possible to perform the registration process with a level of accuracy sufficient for most pelvic cancer sites for rotations up to 17° around z axis, but results for the MV EPID images were not so good because of their low quality (compared to the kV ones). Moreover, the kV/kV results remain acceptable even when rendering sparse DRR images at 10% of ROI. Overall, using kV/kV registration it is possible to recover a movement of $\leq$ 3 mm in the lateral direction , $\leq$ 1 mm in the longitudinal direction, $\leq$ 1 mm in the vertical direction and $< 1^\circ$ in couch return for DRR and EPID images that are rendered in a range of $0^\circ \rightarrow +18^\circ$ rotational movement around z axis.

8.2 Future Research

A number of areas for future research have been identified during this work, these are as follows:

- In our investigation of the hybrid based acceleration for 2D/3D image registration, Section 6.5, we implemented a DRR image rendering approach using GPU hardware. Another design for a hybrid CPU/GPU based system (Figure 6.8) also implemented the similarity metric algorithm using the GPU as a hardware process. This solution is worthy of future investigation since the whole registration process is performed as a hardware process and is potentially more efficient.

- Splat rendering is another method of accelerating DRR rendering (i.e. This technique represent Octree internal spaces by points. The image quality depends on the Octree compression factor. But initial results show that reasonable quality of DRR images can be rendered with highly compressed data). Preliminary results of implementing a splat rendering method show some promising results. We are are able to render DRR images with 256×256 pixels resolution from a $256 \times 256 \times 256$ CT volume at a speed of $\sim$ 330 fps using a DELL XPS Quad core, Intel® 2.3GHz with NVidia GeForce 8800 GT. Figure 8.1 shows an example DRR image rendered using splat rendering. However, measuring the quality of DRR images and the accuracy for 2D/3D image registration remain tasks for future work.

- There is a high demand for fast Tomotherapy KVCT/MVCT image registration [78]. We believe that our method of using sparse images for 2D/3D registration will improve the speed for Tomotherapy registration with acceptable levels of accuracy. However, these assumptions need to be confirmed by experimentation.

- In the European Congress of Radiology (ECR) 2007, Dr. Bordy said "the number one killer disease in the US is not cancer or heart diseases but variability in care". Based on this view, measuring the clinical performance for different image registration techniques could be another area for future

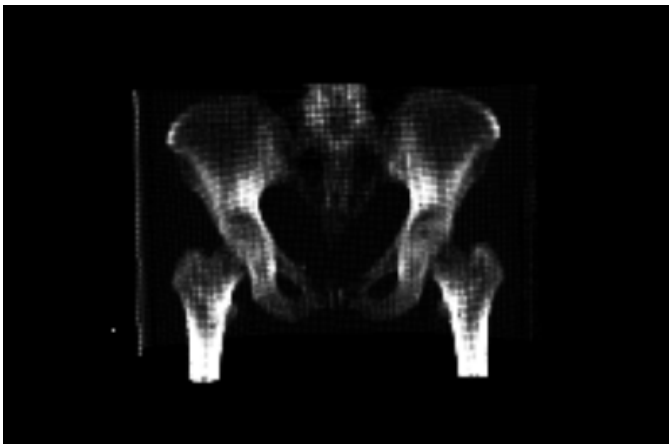


Figure 8.1: An example of DRR image rendered using splat rendering with OpenGL Vertex Buffer Object (VBO) and Octree partitions with Pivot value of 0.2 (more details about Octree partitioning are presented in Section 4.3)

research. In the case of radiation therapy, the uncertainty in the delivery of radiation therapy could go some way towards explaining the variability in treatment outcomes. Further clinical work is needed to explore more the performance of 2D/3D registration within radio oncology.

Appendix A

Supplementary Results

	Misalignment			
	Vertical (cm)	**Longitudinal (cm)**	**Lateral (cm)**	**Couch Return (deg)**
Angle 0	0	0.1	-0.2	0.7
Angle 1	0	0.1	-0.2	0.8
Angle 2	0	0.1	-0.2	0.2
Angle 3	0	0.1	-0.2	0.3
Angle 4	0	0.1	-0.2	0.3
Angle 5	0	0.1	-0.3	0.3
Angle 6	0	0.1	-0.3	0.2
Angle 7	0	0.1	-0.3	0.3
Angle 8	0	0.1	-0.3	0.2
Angle 9	0	0.1	-0.3	0.8
Angle 10	-0.1	0.1	-0.3	0.5
Angle 11	-0.1	0.1	-0.3	0.5
Angle 12	-0.1	0.1	-0.3	0.2
Angle 13	-0.1	0.1	-0.3	0.5
Angle 14	-0.1	0.1	-0.3	0.6
Angle 15	-0.1	0.1	-0.3	0.1
Angle 16	-0.1	0.1	-0.3	0.1
Angle 17	-0.1	0.1	-0.3	0.7
Angle 18	-0.1	0.1	-0.3	0.1
Angle 19	-0.1	0.2	-0.4	0.7
Angle 20	-0.1	0.2	-0.4	0.1

Table A.1: Directional error of kV (DRR/EPID) images using NNUH Varian commercial software.

		DRR Image										
	Angle	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Reference Image	**0**	**0.944995**	0.94445	0.942889	0.940388	0.93688	0.933863	0.929801	0.926322	0.922219	0.918975	0.915166
	1	0.944842	**0.945744**	0.94559	0.944058	0.941368	0.938869	0.935046	0.931825	0.927924	0.924848	0.921176
	2	0.942454	0.944298	**0.945637**	0.945374	0.943668	0.941898	0.938373	0.935409	0.93163	0.928692	0.925089
	3	0.940376	0.942731	0.945071	**0.946289**	0.945963	0.945181	0.942239	0.939625	0.936008	0.933071	0.929472
	4	0.937907	0.940684	0.943512	0.94564	**0.946829**	0.947323	0.945198	0.943245	0.940035	0.937273	0.933733
	5	0.934862	0.937885	0.941117	0.943789	0.945952	**0.947879**	0.946842	0.945766	0.943131	0.940682	0.937251
	6	0.931985	0.935223	0.938642	0.941733	0.944446	0.947314	**0.947628**	0.947798	0.946071	0.944212	0.941166
	7	0.928823	0.932163	0.935669	0.938933	0.942017	0.945424	0.946628	**0.948167**	0.947698	0.946658	0.944176
	8	0.925667	0.929112	0.932709	0.936014	0.939174	0.942893	0.944597	0.946949	**0.947864**	0.947973	0.946276
	9	0.922912	0.926411	0.930066	0.933291	0.93648	0.940335	0.942322	0.945094	0.946885	**0.948251**	0.947699
	10	0.921921	0.925343	0.928943	0.932056	0.935073	0.938773	0.940796	0.943656	0.945729	0.94773	**0.948463**
	11	0.919091	0.922597	0.926339	0.929445	0.932541	0.936232	0.938239	0.941155	0.943565	0.945882	0.947412
	12	0.91632	0.919936	0.923843	0.926997	0.930213	0.933931	0.935955	0.938788	0.941252	0.943795	0.945737
	13	0.913589	0.917326	0.921289	0.924547	0.927881	0.931697	0.933741	0.936549	0.93907	0.941624	0.943764
	14	0.910135	0.913866	0.917966	0.921396	0.924916	0.928873	0.931008	0.933869	0.936427	0.938937	0.941149
	15	0.907086	0.910868	0.91503	0.918577	0.922249	0.926326	0.928518	0.931431	0.934109	0.936638	0.938786
	16	0.903819	0.90772	0.911951	0.915595	0.919317	0.923521	0.925877	0.928871	0.931684	0.934263	0.936424
	17	0.899381	0.903425	0.907773	0.911569	0.915401	0.919711	0.9223	0.925477	0.928424	0.931112	0.93334
	18	0.895101	0.899158	0.903579	0.907484	0.911372	0.91571	0.918408	0.921808	0.924964	0.927814	0.930187
	19	0.891168	0.895343	0.899803	0.903769	0.907729	0.912137	0.914978	0.918454	0.921719	0.924738	0.927181
	20	0.885828	0.890091	0.894553	0.898609	0.90268	0.907113	0.910112	0.913681	0.917071	0.92027	0.922914

Table A.2: Results of performing 2D/3D registration process (NCC values) between kV reference image and FR-DRR images in range of $0° \rightarrow +20°$.

Reference Image	DRR Image									
Angle	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	0.911699	0.908071	0.904405	0.900333	0.896412	0.892829	0.889075	0.88633	0.881485	0.870216
1	0.917922	0.914474	0.911022	0.907034	0.903089	0.899505	0.895672	0.892983	0.888313	0.877314
2	0.921901	0.918548	0.915273	0.911265	0.907297	0.903699	0.899764	0.897173	0.893149	0.883282
3	0.926317	0.923022	0.919875	0.916048	0.912177	0.908624	0.904636	0.902	0.897852	0.88764
4	0.930662	0.92743	0.924378	0.920612	0.916769	0.913223	0.90916	0.90655	0.902577	0.892857
5	0.934213	0.931064	0.928161	0.924602	0.920939	0.917538	0.913484	0.91067	0.906161	0.89633
6	0.938311	0.935232	0.932396	0.928912	0.925268	0.921853	0.917762	0.914915	0.910071	0.899609
7	0.941705	0.938774	0.935952	0.932439	0.928887	0.925543	0.921501	0.918461	0.913484	0.902613
8	0.944356	0.941737	0.93903	0.935546	0.932037	0.928742	0.924696	0.921714	0.916259	0.90506
9	0.946603	0.944428	0.941902	0.938503	0.934949	0.931641	0.927646	0.92471	0.919251	0.907685
10	0.948397	0.946772	0.944581	0.941305	0.937789	0.934388	0.930304	0.927296	0.921718	0.909865
11	**0.948639**	0.947974	0.946448	0.943552	0.940205	0.936858	0.932823	0.929677	0.923749	0.911431
12	0.947673	**0.94813**	0.947697	0.945461	0.942535	0.939369	0.935417	0.93234	0.926081	0.913262
13	0.946014	0.94713	**0.948014**	0.946756	0.944382	0.941535	0.937841	0.934872	0.928666	0.915672
14	0.943651	0.945166	0.946825	**0.946821**	0.945477	0.943289	0.939995	0.937036	0.93012	0.916352
15	0.941351	0.943058	0.945105	0.945889	**0.945702**	0.944584	0.941909	0.939127	0.931712	0.917507
16	0.938924	0.940752	0.943038	0.944241	0.944845	**0.944947**	0.943235	0.940859	0.933046	**0.918045**
17	0.93589	0.937826	0.940298	0.94182	0.942976	0.943924	**0.943428**	0.941768	0.932984	0.916905
18	0.932872	0.934944	0.937557	0.939297	0.94089	0.942401	0.942801	**0.942378**	**0.933721**	0.916572
19	0.929948	0.932078	0.934723	0.936539	0.938388	0.940263	0.94115	0.94122	0.932575	0.914979
20	0.92586	0.928162	0.930944	0.932935	0.934946	0.937078	0.938416	0.938876	0.929682	0.911278

Table A.3: Cont. to Table A.2

Reference Image	DRR Image										
Angle	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	**0.944893**	0.944359	0.942791	0.940281	0.936758	0.933745	0.929665	0.926186	0.922081	0.918828	0.915019
1	0.944618	**0.945503**	0.945369	0.943878	0.941212	0.938769	0.935019	0.931814	0.927942	0.92492	0.921277
2	0.94259	0.944432	**0.945757**	0.945525	0.943877	0.942154	0.938647	0.935723	0.932006	0.929074	0.925477
3	0.940325	0.942638	0.944947	**0.946136**	0.945773	0.945008	0.942058	0.939407	0.935766	0.932867	0.929262
4	0.937793	0.940577	0.943459	0.945613	**0.946849**	0.947408	0.945305	0.94341	0.940186	0.937406	0.933872
5	0.934748	0.937775	0.941027	0.943674	0.945826	**0.947783**	0.94676	0.945675	0.943001	0.940525	0.937068
6	0.931755	0.935016	0.93843	0.94153	0.944258	0.947185	**0.947506**	0.947668	0.945972	0.944163	0.941143
7	0.928612	0.931951	0.93545	0.938712	0.941785	0.945142	0.946353	**0.947899**	0.947376	0.946295	0.943813
8	0.925342	0.92879	0.932376	0.93567	0.938862	0.94257	0.944292	0.94665	**0.947561**	0.947677	0.945999
9	0.922231	0.925781	0.929466	0.932727	0.935927	0.939797	0.941822	0.944611	0.94641	**0.947795**	0.947261
10	0.921127	0.924572	0.928245	0.93141	0.934489	0.938233	0.940293	0.943208	0.945357	0.947431	**0.948184**
11	0.918485	0.92205	0.925815	0.928948	0.932084	0.935835	0.937873	0.940848	0.943303	0.945661	0.94721
12	0.915561	0.919216	0.923182	0.926384	0.929679	0.933464	0.935517	0.938414	0.94093	0.943534	0.945517
13	0.913078	0.916818	0.920848	0.924147	0.927544	0.931396	0.933488	0.936369	0.938897	0.94147	0.943628
14	0.909133	0.9129	0.917064	0.920565	0.924147	0.92816	0.930336	0.933277	0.9359	0.938479	0.940769
15	0.90655	0.910374	0.914529	0.918132	0.921869	0.925976	0.928222	0.931184	0.933914	0.936471	0.938674
16	0.903229	0.907141	0.911359	0.915029	0.91878	0.923009	0.925372	0.92842	0.931305	0.933929	0.936127
17	0.898875	0.902904	0.907224	0.91102	0.914845	0.919136	0.921729	0.924934	0.927939	0.930665	0.932945
18	0.894494	0.898556	0.902969	0.906866	0.910781	0.915116	0.917826	0.921266	0.924441	0.927317	0.929702
19	0.890623	0.894812	0.899259	0.903213	0.907181	0.911577	0.914402	0.917886	0.921149	0.924183	0.926647
20	0.884924	0.889192	0.893672	0.897734	0.901818	0.906256	0.909281	0.912898	0.916313	0.919535	0.922196

Table A.4: Results of performing 2D/3D registration process (NCC values) between kV reference image and RR-DRR images (50% sampling) in range of $0° \rightarrow +20°$.

		DRR Image									
Reference Image	**Angle**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	0	0.911699	0.908071	0.904405	0.900333	0.896412	0.892829	0.889075	0.88633	0.881485	0.870216
	0	0.91153	0.907914	0.904252	0.900148	0.896214	0.892619	0.888847	0.886109	0.8814	0.870288
	1	0.91808	0.914684	0.911255	0.907321	0.903416	0.899863	0.896048	0.893362	0.888615	0.877544
	2	0.922279	0.91894	0.915688	0.911709	0.907777	0.9042	0.900246	0.897691	0.893697	0.883802
	3	0.926161	0.922873	0.919748	0.915935	0.912057	0.908518	0.904545	0.901917	0.897749	0.887602
	4	0.930766	0.927528	0.924483	0.920758	0.91696	0.913451	0.909414	0.906779	0.90267	0.89281
	5	0.934054	0.930911	0.928005	0.924459	0.920814	0.917419	0.913349	0.910461	0.905992	0.896167
	6	0.938306	0.93524	0.932423	0.928931	0.925279	0.921841	0.917756	0.91488	0.909839	0.899268
	7	0.941309	0.938335	0.935486	0.931979	0.928385	0.925037	0.920976	0.918022	0.913179	0.902455
	8	0.944103	0.941473	0.938772	0.935298	0.931794	0.928498	0.924455	0.921483	0.916006	0.904855
	9	0.946167	0.943992	0.941472	0.938102	0.934553	0.931267	0.927261	0.924303	0.918739	0.907177
	10	0.948138	0.946513	0.944357	0.941112	0.937589	0.934185	0.930146	0.927197	0.921667	0.909941
	11	**0.948489**	0.947863	0.946375	0.943479	0.940164	0.936831	0.932833	0.92969	0.923807	0.911472
	12	0.947474	**0.948005**	0.94763	0.945426	0.942543	0.939356	0.935416	0.93236	0.926166	0.91331
	13	0.945934	0.947104	**0.948024**	**0.946791**	0.94445	0.941654	0.937989	0.935097	0.928978	0.916024
	14	0.943344	0.944929	0.946674	0.946734	0.945463	0.943337	0.940077	0.937142	0.930125	0.916167
	15	0.94127	0.943027	0.945147	0.945977	**0.945859**	0.944801	0.942162	0.939407	0.931872	0.917469
	16	0.938677	0.940559	0.942898	0.944122	0.944767	**0.944948**	0.943285	0.940927	0.933249	**0.918205**
	17	0.935548	0.937535	0.940052	0.941603	0.942812	0.943806	**0.943346**	0.941729	0.932845	0.916597
	18	0.932466	0.934543	0.937201	0.938978	0.940622	0.942181	0.942644	**0.942183**	**0.933428**	0.916154
	19	0.929472	0.931633	0.934326	0.936203	0.938074	0.939975	0.940902	0.940977	0.932268	0.914638
	20	0.925178	0.927533	0.930365	0.932408	0.934445	0.936648	0.938038	0.93856	0.929414	0.910966

Table A.5: Cont. to Table A.4

Reference Image	DRR Image										
Angle	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0.940707	0.940253	0.938761	0.936348	0.932863	0.929728	0.925571	0.921987	0.917767	0.914344	0.910462
1	**0.943522**	**0.944403**	**0.944178**	0.942577	0.939836	0.937337	0.933509	0.930177	0.926209	0.923028	0.919329
2	0.939865	0.942222	0.943908	**0.944015**	0.942553	0.940965	0.937659	0.934975	0.931487	0.928832	0.925452
3	0.937977	0.940072	0.942159	0.943201	0.942723	0.9418	0.938825	0.936137	0.932605	0.92969	0.926171
4	0.934999	0.937795	0.940704	0.942905	**0.944111**	0.9446	0.942433	0.940491	0.937372	0.934787	0.931424
5	0.933228	0.936116	0.939185	0.941773	0.943861	0.945807	0.944746	0.943678	0.941131	0.938884	0.935579
6	0.931151	0.934349	0.937695	0.940675	0.943296	**0.946137**	**0.946332**	**0.946331**	0.944604	0.942744	0.939713
7	0.926738	0.929994	0.933425	0.936601	0.93957	0.94297	0.944146	0.945694	**0.945143**	0.944047	0.941453
8	0.920309	0.923869	0.927616	0.931098	0.934491	0.938364	0.940175	0.94273	0.943724	0.943937	0.942266
9	0.917109	0.921014	0.924958	0.928478	0.931892	0.935906	0.938048	0.940962	0.942957	0.944403	0.943964
10	0.918597	0.92208	0.925837	0.929068	0.932228	0.93609	0.938233	0.941308	0.943519	**0.945633**	**0.946428**
11	0.915408	0.919109	0.923062	0.926372	0.929708	0.933659	0.935826	0.93898	0.941442	0.943945	0.945622
12	0.913233	0.917018	0.921026	0.924294	0.927702	0.931571	0.933734	0.936825	0.939432	0.942016	0.943978
13	0.910304	0.913988	0.917893	0.921224	0.924726	0.928549	0.930737	0.93382	0.936455	0.939168	0.941393
14	0.906266	0.90994	0.914018	0.917466	0.921049	0.925144	0.927313	0.930274	0.93292	0.935549	0.937861
15	0.901433	0.905344	0.909649	0.913506	0.917334	0.921684	0.924155	0.927303	0.93017	0.932852	0.935209
16	0.900644	0.904576	0.908889	0.912635	0.916596	0.921027	0.923586	0.926896	0.929898	0.932597	0.93484
17	0.895931	0.899867	0.904181	0.907969	0.911804	0.916204	0.918882	0.922337	0.925432	0.928287	0.930585
18	0.88959	0.893682	0.898116	0.902011	0.906024	0.91044	0.91327	0.916843	0.920192	0.923217	0.925663
19	0.888937	0.893049	0.897408	0.901301	0.905361	0.90977	0.912672	0.916285	0.919632	0.922704	0.925159
20	0.884757	0.889013	0.89343	0.897479	0.901641	0.906195	0.909168	0.912747	0.916159	0.919272	0.92189

Table A.6: Results of performing 2D/3D registration process (NCC values) between kV reference image and RR-DRR images (75% sampling) in range of $0° \rightarrow +20°$.

Reference Image	DRR Image									
Angle	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	0.906807	0.903129	0.899406	0.895253	0.891229	0.88751	0.883587	0.880626	0.875783	0.865306
1	0.915914	0.912381	0.908737	0.904656	0.900648	0.896991	0.892994	0.889974	0.885243	0.874652
2	0.922492	0.919368	0.916283	0.912497	0.908689	0.905255	0.901366	0.89904	0.895273	0.884278
3	0.923139	0.919928	0.916891	0.913098	0.909223	0.905597	0.901598	0.898718	0.893342	0.883029
4	0.928449	0.925342	0.922432	0.918802	0.915032	0.911623	0.90752	0.904867	0.90148	0.8925
5	0.932599	0.92955	0.926696	0.923187	0.919556	0.916152	0.91214	0.909549	0.905818	0.896538
6	0.936917	0.933898	0.931065	0.927705	0.924079	0.920752	0.916729	0.913778	0.908694	0.897647
7	0.938902	0.935897	0.932958	0.929347	0.925777	0.922482	0.918466	0.915464	0.910805	0.901171
8	0.940493	0.937949	0.935275	0.931829	0.928358	0.925216	0.92129	0.918591	0.913777	0.902729
9	0.942961	0.940921	0.93856	0.935128	0.931556	0.92832	0.924337	0.921248	0.915197	0.902774
10	0.946441	0.944947	0.94298	0.939841	0.936338	0.933035	0.92898	0.926097	0.920961	0.909715
11	**0.947003**	0.946467	0.945189	0.942537	0.939311	0.936165	0.932236	0.929167	0.922531	0.909484
12	0.945995	**0.946515**	**0.946188**	0.944022	0.941123	0.938055	0.93413	0.93126	0.925249	0.913148
13	0.943834	0.945025	0.945969	**0.944778**	0.942451	0.939589	0.93582	0.932891	0.926403	0.913154
14	0.940503	0.942153	0.943922	0.944073	0.942875	0.940831	0.93757	0.934598	0.927035	0.913082
15	0.937866	0.939745	0.941992	0.943035	0.943194	0.942348	0.939877	0.937479	0.929932	0.915783
16	0.937504	0.939347	0.941722	0.943033	**0.943859**	**0.94417**	**0.942591**	**0.940451**	**0.932092**	**0.916942**
17	0.933244	0.935154	0.937654	0.939211	0.940403	0.94143	0.941016	0.939735	0.930979	0.915862
18	0.928497	0.930585	0.933316	0.93523	0.936946	0.938744	0.939417	0.93936	0.930069	0.912276
19	0.928081	0.930245	0.932841	0.934699	0.936588	0.938449	0.939358	0.939572	0.930448	0.913575
20	0.924773	0.927042	0.929763	0.931552	0.933596	0.935815	0.93725	0.938142	0.930021	0.912445

Table A.7: Cont. to Table A.6

		DRR Image										
	Angle	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Reference Image	**0**	0.798769	0.798384	0.79726	0.795021	0.792289	0.789494	0.786281	0.783676	0.780061	0.777398	0.774052
	1	**0.799135**	**0.799782**	0.799625	0.798124	0.796054	0.79372	0.790696	0.788354	0.784923	0.782434	0.779243
	2	0.79758	0.798974	**0.799798**	0.799259	0.797887	0.796134	0.793419	0.791309	0.788095	0.785743	0.782644
	3	0.796022	0.798047	0.799781	**0.800257**	0.799905	0.798856	0.796615	0.794796	0.791646	0.789385	0.78631
	4	0.793786	0.796207	0.798554	0.799821	**0.800465**	0.800398	0.798743	0.797515	0.794776	0.792645	0.789668
	5	0.790923	0.793654	0.796439	0.798288	0.799807	**0.800703**	**0.7998**	0.799105	0.796786	0.794948	0.792097
	6	0.78772	0.790671	0.793692	0.79595	0.798112	0.79973	0.799708	0.799937	0.798281	0.796916	0.794427
	7	0.784889	0.788026	0.791136	0.793628	0.796176	0.798423	0.799139	**0.800267**	0.799547	0.798752	0.796666
	8	0.782376	0.785555	0.788803	0.791352	0.794042	0.796723	0.798025	0.799888	0.800061	0.800074	0.798518
	9	0.780573	0.78379	0.787076	0.789637	0.792412	0.79525	0.796936	0.799349	**0.800251**	**0.801081**	0.800357
	10	0.779415	0.782611	0.785922	0.788442	0.791099	0.793842	0.7956	0.7982	0.799525	0.800891	0.800963
	11	0.77809	0.781316	0.784686	0.787209	0.789943	0.792703	0.79451	0.79725	0.798954	0.800768	**0.80153**
	12	0.775803	0.779145	0.782645	0.785149	0.787917	0.790744	0.792584	0.795239	0.797068	0.799186	0.800481
	13	0.773553	0.77696	0.780536	0.7831	0.785969	0.788879	0.790773	0.793486	0.795417	0.79762	0.799198
	14	0.771078	0.774534	0.77823	0.780942	0.783971	0.786957	0.788909	0.791688	0.793684	0.795906	0.797669
	15	0.769577	0.773084	0.776837	0.779634	0.782819	0.785879	0.787825	0.790574	0.792648	0.794894	0.796605
	16	0.767674	0.771269	0.775046	0.777912	0.781112	0.784332	0.786428	0.789213	0.791385	0.793666	0.795455
	17	0.765098	0.768848	0.772734	0.775785	0.779124	0.782432	0.784728	0.787745	0.790017	0.792368	0.794182
	18	0.762606	0.766383	0.770338	0.773442	0.77682	0.780184	0.782586	0.785801	0.788363	0.790916	0.792863
	19	0.761141	0.764994	0.768984	0.772185	0.775637	0.77908	0.781535	0.784706	0.787303	0.789949	0.791905
	20	0.758769	0.762724	0.766699	0.77	0.773603	0.777046	0.779712	0.783048	0.785761	0.788568	0.790726

Table A.8: Results of performing 2D/3D registration process (NCC values) between kV reference image and RR-DRR images (75% rays reduction) in range of $0° \rightarrow +20°$.

Reference Image	DRR Image									
Angle	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
0	0.771017	0.767537	0.764641	0.761019	0.757454	0.754456	0.751144	0.74851	0.744888	0.734948
1	0.776383	0.773052	0.77025	0.766643	0.763079	0.76006	0.756676	0.754059	0.750638	0.741131
2	0.779905	0.776709	0.774134	0.770517	0.766945	0.763921	0.76051	0.758032	0.755091	0.746503
3	0.783616	0.78048	0.778003	0.774473	0.770864	0.767843	0.764359	0.76183	0.758774	0.749919
4	0.787014	0.783949	0.781541	0.778126	0.774649	0.771656	0.768176	0.765558	0.762374	0.753948
5	0.789588	0.78663	0.784411	0.78119	0.77786	0.77505	0.771595	0.768919	0.765211	0.756531
6	0.792121	0.789285	0.787159	0.783959	0.780693	0.777881	0.774429	0.771744	0.767675	0.758172
7	0.794693	0.792003	0.789887	0.786776	0.783596	0.780874	0.77744	0.774679	0.770895	0.76112
8	0.796964	0.794558	0.792585	0.789494	0.786336	0.783656	0.780223	0.777546	0.77331	0.763486
9	0.799376	0.797356	0.795549	0.792604	0.789444	0.786753	0.783379	0.780735	0.77629	0.765995
10	0.800727	0.799095	0.797546	0.79466	0.791605	0.788896	0.785502	0.782737	0.778107	0.767517
11	**0.802039**	0.801073	0.799983	0.797462	0.794486	0.79182	0.788477	0.785723	0.781056	0.770297
12	0.801623	**0.801305**	0.80093	0.798839	0.796194	0.793661	0.79038	0.787593	0.782656	0.77122
13	0.800802	0.801092	0.801533	0.800116	0.797862	0.795599	0.79251	0.789989	0.784819	0.773484
14	0.799602	0.800344	**0.801561**	0.80096	0.799423	0.797658	0.794902	0.792192	0.78676	0.774842
15	0.798585	0.799585	0.801216	**0.801258**	0.800454	0.799327	0.796996	0.794474	0.788437	0.776286
16	0.797457	0.798583	0.800537	0.801099	0.800928	0.800591	0.7989	0.796711	0.790568	0.777523
17	0.796285	0.797592	0.799744	0.800665	**0.801102**	0.801517	0.800634	0.798915	0.791873	0.778089
18	0.795131	0.796614	0.7989	0.800074	0.800943	0.801988	0.801845	0.800908	0.793606	0.778585
19	0.794267	0.795772	0.798184	0.799435	0.800613	**0.802064**	**0.802489**	0.801916	0.794482	**0.778854**
20	0.793239	0.794943	0.797454	0.7989	0.800254	0.801964	0.802832	**0.802725**	**0.794634**	0.778018

Table A.9: Cont. to Table A.8

	Best Match of DRR Images			
Reference Image	**FR-DRR**	**RR-DRR 50% (using sampling)**	**RR-DRR 75% (using sampling)**	**RR-DRR 75% (using rays reduction)**
0	0	0	1	1
1	1	1	1	1
2	2	2	1	2
3	3	3	2	3
4	4	4	4	4
5	5	5	6	5
6	6	6	6	5
7	7	7	6	7
8	8	8	7	9
9	9	9	10	9
10	10	10	10	11
11	11	11	11	11
12	12	12	12	12
13	13	13	12	14
14	14	13	13	15
15	15	15	16	17
16	16	16	16	19
17	17	17	16	19
18	18	18	16	20
19	18	18	16	20
20	16	16	16	19

Table A.10: Accuracy for the performance of the 2D/3D registration using different methods of rendering the DRR images (full and reduced resolution images).

***Rays Reduction* Method of RR-DRR**	Number of Cores (Threads)							
	Single		Dual		Quad		Octal	
Image Direction / CT Volume Size	0°	90°	0°	90°	0°	90°	0°	90°
$128 \times 128 \times 66$ (pelvis)	29 ms	26 ms	16 ms	13 ms	15 ms	12 ms	12 ms	6 ms
$128 \times 128 \times 86$ (lung)	35 ms	35 ms	18 ms	17 ms	16 ms	15 ms	8 ms	8 ms
$256 \times 256 \times 133$ (pelvis)	293 ms	242 ms	141 ms	114 ms	102 ms	97 ms	58 ms	49 ms
$256 \times 256 \times 172$ (lung)	333 ms	278 ms	205 ms	158 ms	144 ms	115 ms	75 ms	64 ms
$512 \times 512 \times 267$ (pelvis)	2051 ms	1586 ms	1527 ms	1291 ms	783 ms	677 ms	444 ms	369 ms
$512 \times 512 \times 344$ (lung)	5915 ms	2008 ms	1880 ms	1083 ms	1056 ms	903 ms	560 ms	478 ms

Table A.11: Results in milliseconds of running the parallelised algorithm of rendering reduced resolution DRR images using different number of cores and different sizes of CT volumes (pelvis and lung), DRR images were rendered from y direction at 0° and 90° with reduction in the number of x-rays by 25% of total number of x-rays which used to render FR-DRR image using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz.

***Sampling* Method of RR-DRR**	Number of Cores (Threads)							
	Single		Dual		Quad		Octal	
Sampling reduction / CT Volume Size	50%	75%	50%	75%	50%	75%	50%	75%
$128 \times 128 \times 66$ (pelvis)	53 ms	28 ms	28 ms	15 ms	25 ms	13 ms	13 ms	8 ms
$128 \times 128 \times 86$ (lung)	72 ms	37 ms	37 ms	19 ms	32 ms	17 ms	19 ms	9 ms
$256 \times 256 \times 133$ (pelvis)	429 ms	219 ms	230 ms	121 ms	194 ms	95 ms	100 ms	51 ms
$256 \times 256 \times 172$ (lung)	573 ms	296 ms	299 ms	152 ms	255 ms	125 ms	130 ms	66 ms
$512 \times 512 \times 267$ (pelvis)	3153 ms	1691 ms	2162 ms	1066 ms	1359 ms	709 ms	750 ms	391 ms
$512 \times 512 \times 344$ (lung)	4105 ms	2201 ms	2142 ms	1322 ms	1742 ms	921 ms	973 ms	510 ms

Table A.12: Results in milliseconds of running the parallelised algorithm of rendering reduced resolution DRR images using different number of cores and different sizes of CT volumes (pelvis and lung), DRR images were rendered with reduction in the sample points by 50% and 75% of the FR-DRR image using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz.

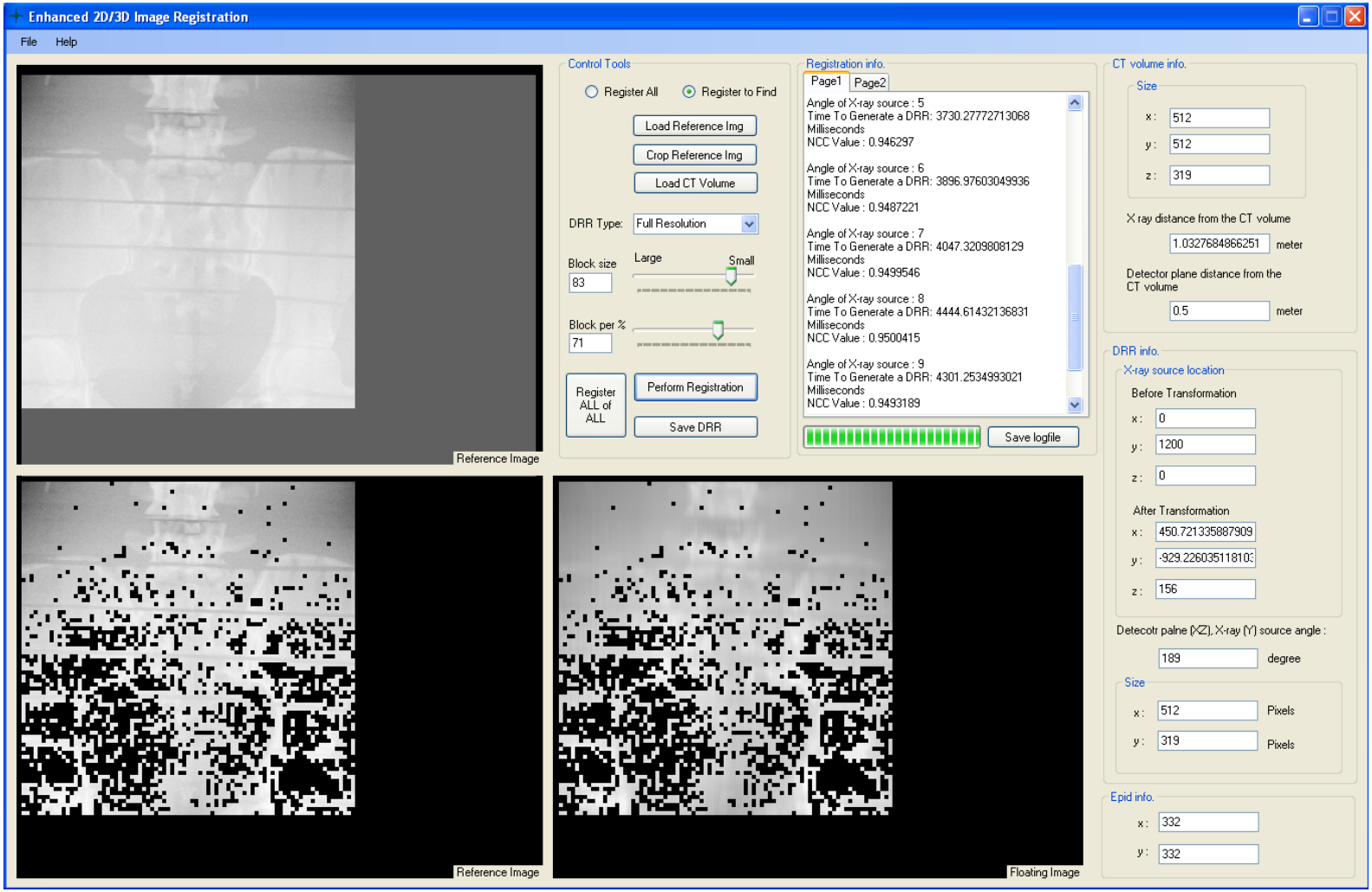


Figure A.1: GUI combined our developed methods for enhancing the speed 2D/3D image registration system.

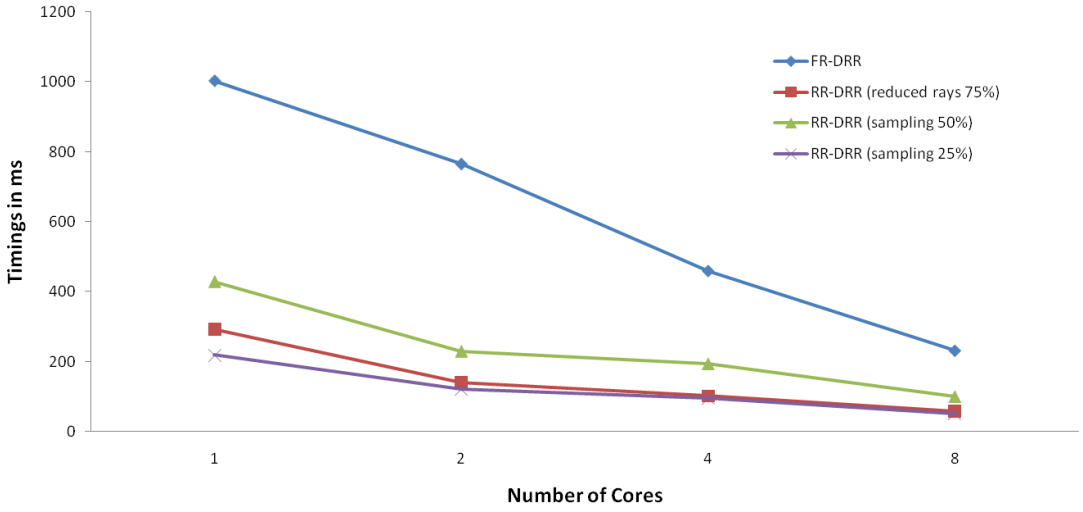


Figure A.2: Results in milliseconds of rendering different types of DRR images (full resolution and reduced resolution) from $256 \times 256 \times 133$ pelvis CT volume using the parallelised algorithm of DRR rendering and using a Precision Workstation T5500 Dual Quad core, Intel® 2.3GHz.

Appendix B

Ray-box Intersection Algorithm

Algorithm 4 RayBox Intersection Algorithm [8]

```
1: Vector3 = (float x, float y, float z)
2: Vector3 parameters[0] = min_box_3dCoordinates
3: Vector3 parameters[1] = max_box_3dCoordinates
4: Vector3 origin
5: Vector3 direction
6: Vector3 inv_direction
7: inv_direction = (1/direction.x(), 1/direction.y(), 1/direction.z())
8: integer sign[3]
9: sign[0] = boolean (inv_direction.x() < 0)
10: sign[1] = boolean (inv_direction.y() < 0)
11: sign[2] = boolean (inv_direction.z() < 0)
12: boolean Box_intersect(Ray r, float tIn, float tOut)
13: {
14: float tmin, tmax, tymin, tymax, tzmin, tzmax
15: tmin = (parameters[r.sign[0]].x - r.origin.x)  × r.inv_direction.x
16: tmax = (parameters[1-r.sign[0]].x - r.origin.x)  × r.inv_direction.x
17: tymin = (parameters[r.sign[1]].y - r.origin.y)  × r.inv_direction.y
18: tymax = (parameters[1-r.sign[1]].y - r.origin.y)  × r.inv_direction.y
19: if (tmin > tymax) or (tymin > tmax)  then
20:     return  false
21: end if
22: if (tymin  > tmin)  then
23:     tmin = tymin
24: end if
25: if (tymax  < tmax)  then
26:     tmax = tymax
27: end if
28: tzmin = (parameters[r.sign[2]].z - r.origin.z) × r.inv_direction.z
29: tzmax = (parameters[1-r.sign[2]].z - r.origin.z) × r.inv_direction.z
30: if (tmin > tzmax) or (tzmin > tmax)  then
31:     return  false
32: end if
33: if  (tzmin > tmin) then
34:     tmin = tzmin
35: end if
36: if (tzmax < tmax)  then
37:     tmax = tzmax
38: end if
39: return  false
40: }
```

Bibliography

[1] Nobel Web AB. X-rays, what are they? http://nobelprize.org/educational/physics/x-rays/what-3.html, August, 2010.

[2] ACCURAY. Cyberknife system. http://www.accuray.com/products/CyberKnife-System.aspx, 23 September, 2010.

[3] The National Aeronautics and Space Administration (NASA). X-rays. http://science.hq.nasa.gov/kids/imagers/ems/xrays.html, August, 2010.

[4] Sectio Ai and Przemysaw Stpiczyski. Ada as a language for programming clusters of smps. *Annales UMCS Informatica AI 1*, 1:73–79, December 1988.

[5] Oleg Alexandrov (Wikipedia). Gradient descent. http://edndoc.esri.com/arcobjects/9.2/net/b0e91ce8-c180-47dc-8323-06cac5d77064.htm, August, 2010.

[6] M.L.G. Althouse and Chein-I Chang. Image segmentation by local entropy methods. *Image Processing, International Conference on*, 3:3061, 1995.

[7] John Amanatides and Andrew Woo. A fast voxel traversal algorithm for ray tracing. In *In Eurographics 87*, pages 3–10, 1987.

[8] Williams Amy, Barrus Steve, R. Keith Morley, and Shirley Peter. An efficient and robust ray-box intersection algorithm, 2005.

[9] Limin Luo Aodong Shen. Point-based digitally reconstructed radiograph. In *ICPR08*, pages 1–4, 2008.

[10] J. Ashburner and K.J. Friston. Rigid body registration. In R.S.J. Frackowiak, K.J. Friston, C. Frith, R. Dolan, K.J. Friston, C.J. Price, S. Zeki, J. Ashburner, and W.D. Penny, editors, *Human Brain Function*. Academic Press, second edition, 2003.

[11] Ravi Bansal, Lawrence H. Staib, and Zhe Chen et.al. *A Novel Approach for the Registration of 2D Portal and 3D CT Images for Treatment Setup Verification in Radiotherapy*. Medical Image Computing and Computer-Assisted Intervention. Springer Berlin / Heidelberg, 1998.

[12] Chapman Barbara, Jost Gabriele, and Pas Ruud van der. *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007. 1370966.

[13] Christopher Bethune and A. James Stewart. Accelerated computation of digitally reconstructed radiographs. *International Congress Series*, 1281:98–103, 2005.

[14] Imma Boada, Isabel Navazo, and Roberto Scopigno. Multiresolution volume visualization with a texture-based octree. *The Visual Computer*, 17:185–197, 2001.

[15] Mario Botsch and Leif Kobbelt. High-quality point-based rendering on modern GPUs. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, PG '03, pages 335–, Washington, DC, USA, 2003. IEEE Computer Society.

[16] Boyd, P. D, Lipton, and J. M. *Cardiac computed tomography*, volume 71. Institute of Electrical and Electronics Engineers, New York, NY, ETATS-UNIS, 1983. 3 Proceedings of the IEEE.

[17] John F. Bruzzi, Reginald F. Munden, Mylene T. Truong, Edith M. Marom, Bradley S. Sabloff, Gregory W. Gladish, Revathy B. Iyer, Tin-Su Pan, Homer A. Macapinlac, and Jeremy J. Erasmus. Pet/ct of esophageal cancer: Its role in clinical management1. *Radiographics*, 27(6):1635–1652, 2007.

[18] A. Burns. *Programming in Occam 2.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1987.

[19] Jerrold T. Bushberg, J.Anthony Seibert, John M. Boone, and Edwin M. Leidholdt. *The Essential Physics of Medical Imaging.* Lippincott Williams and Wilkins, second edition, 2000.

[20] Stewart C. Bushong. *Computed Tomography.* McGraw-Hill Companies, first edition, 2000.

[21] Thorsten M. Buzug. *Computed Tomography: From Photon Statistics to Modern Cone-Beam CT.* Springer, first edition, 2008.

[22] Gibbs I. C. Frameless image-guided intracranial and extracranial radiosurgery using the cyberknife robotic system. *Cancer/Radiotherapie*, 10(5):283–287, 2006. doi: DOI: 10.1016/j.canrad.2006.05.013.

[23] Foran J. Cabral B., Cam N. Accelerated volume rendering and topographic reconstruction using texture mapping hardware. *ACM Symp*, Vol. Vis,:91–98, 1994.

[24] Philippe Cattin. Imaging modalities: X-ray, ct, fluoroscope. http://miac.unibas.ch/BIA/06-Xray.html, 12 April 2010.

[25] Steven D. Chang, William Main, David P. Martin, Iris C. Gibbs, and M. Peter Heilbrun. An analysis of the accuracy of the cyberknife: A robotic frameless stereotactic radiosurgical system. *Neurosurgery*, 52(1):140–147, 2003.

[26] Chin-Tu Chen. Radiologic image registration old skills and new tools. *Academic Radiology*, 10(3):239–241, 2003.

[27] Lili Chen, Jr Robert A. Price, Lu Wang, Jinsheng Li, Lihong Qin, Shawn McNeeley, C. M. Charlie Ma, Gary M. Freedman, and Alan Pollack. Mri-based treatment planning for radiotherapy: Dosimetric verification for prostate imrt. *International Journal of Radiation Oncology*Biology*Physics*, 60(2):636–647, 2004. doi: DOI: 10.1016/j.ijrobp.2004.05.068.

[28] C. H. Chien and J. K. Aggarwal. Identification of 3d objects from multiple silhouettes using quadtrees/octrees. *Computer Vision, Graphics, and Image Processing*, 36(2-3):256–273, 1986. doi: DOI: 10.1016/0734-189X(86)90078-2.

[29] Antypas Christos and Pantelis Evaggelos. *Performance evaluation of a CyberKnife G4 image-guided robotic stereotactic radiosurgery system*, volume 53. Institute of Physics, Bristol, ROYAUME-UNI, 2008. 17 Physics in medicine and biology.

[30] Kevin A. Camphausen Coia and Lawrence R. *Cancer Management: A Multidisciplinary Approach.* New York: CMP Helth Care Media LLC., 11th edition, 2008.

[31] J.D. Cox, W. Ki Hong, J.A. Roth, Frank V. Fossella, Joe B. Jr. Putnam, and Ritsuko Komaki. *Lung Cancer (M.D. Anderson Cancer Care Series).* Springer, 2002.

[32] I. A. Cunningham and P. F. Judy. Computed tomography. In Editor Joseph D. Bronzino, editor, *The Biomedical Engineering Handbook*, volume Vol 1. Springer-Verlag Berlin and Heidelberg GmbH Co. K, second edition, 2000.

[33] Woodford Curtis, Yartsev Slav, and Dyk Jake Van. Image registration of a moving target phantom with helical tomotherapy: effect of the ct acquisition technique and action level proposal. *Physics in Medicine and Biology*, 53:5093–5106, 2008. doi:10.1088/0031-9155/53/18/016.

[34] Slav Yartsev Curtis Woodford and Jake Van Dyk. Optimization of megavoltage ct scan registration settings for thoracic cases on helical tomotherapy. *Physics in Medicine and Biology*, 52(15):N3 45–54, 2007.

[35] G. L. Davies and A. Burns. The teaching language pascal-fc. *Comput. J.*, 33(2):147–154, 1990.

[36] Meagher D.J.R. Geometric modeling using octree encoding. 19(2):129–147, June 1982.

[37] Fu Dongshan and Kuduvalli Gopinath. A fast, accurate, and automatic 2d–3d image registration for image-guided cranial radiosurgery. *Medical Physics*, 35(5):2180–2194, 2008.

[38] Fu Dongshan, Kuduvalli Gopinath, Mitrovic Vladimir, Main William, and Thomson Larry. Automated skull tracking for the cyberknife image-guided radiosurgery system. volume 5744, pages 366–377. SPIE, 2005. Medical Imaging 2005: Visualization, Image-Guided Procedures, and Display 1.

[39] Osama Dorgham, Mark Fisher, and Stephen Laycock. Performance of a 2D/3D image registration system using lossy) compressed X-ray CT. *Annals of the BMVA*, 2009(3):1–11, October 2009.

[40] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Willey & Sons, New Yotk, 1973.

[41] Luther W. Brady Edward C. Halperin, Carlos A. Perez. *Perez and Brady's principles and practice of radiation oncology*. Lippincott Williams and Wilkins, fifth edition, 2008.

[42] Kayvon Fatahalian and Mike Houston. A closer look at gpus. *Commun. ACM*, 51(10):50–57, 2008.

[43] Fathom. Ada Lovelace. http://www.fathom.com/feature/122251/3134_adalovelace_LG.html, July, 2010.

[44] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993.

[45] R.A. Finkel and J.L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Inf.*, 4(1):1–9, 1974.

[46] J. M. Fitzpatrick, J. B. West, and C. R. Maurer. Predicting error in rigid-body point-based registration. *IEEE Transactions on Medical Imaging*, 17(5):694–702, 1998.

[47] Christopher Fox, H. Edwin Romeijn, and James F. Dempsey. Fast voxel and polygon ray-tracing algorithms in intensity modulated radiation therapy treatment planning. *Medical Physics*, 33(5):1364–1371, 2006.

[48] Leonardo Frighetto. Cyberknife frameless stereotactic radiosurgery for spinal lesions: Clinical experience in 125 cases. *Neurosurgery*, 56(5):E1166 10.1227/01.NEU.0000155100.38862.A6, 2005.

[49] Roland Göcke, Jürgen Weese, and Heidrum Schumann. Fast volume rendering methods for voxel-based 2D/3D registration - A comparative study. In *Proc. 1st Int. Workshop on Biomedial Image Registration*, pages 89–102, Beld, Slovenia, August 1999.

[50] Eddins Gonzalez, Woods. *Digital Image Processing Using MATLAB*. Gatesmark Publishing, second edition, 2009.

[51] Markus Grabner, Thomas Pock, Tobias Gross, and Bernhard Kainz. Automatic differentiation for GPU-accelerated 2D/3D registration. In Christian H. Bischof, H. Martin Bücker, Paul D. Hovland, Uwe Naumann, and J. Utke, editors, *Advances in Automatic Differentiation*, pages 259–269. Springer, 2008.

[52] Booch Grady and Bryan Doug. *Software engineering with Ada (3rd ed.)*. Benjamin-Cummings Publishing Co., Inc., 1993. 162258.

[53] A. O. Griewank. Generalized descent for global optimization. *Journal of Optimization Theory and Applications*, 34(1):11–39, 1981.

[54] Tobias Gross. GPU-Based 2D/3D registration of X-ray and CT data. Master's thesis, Institute for Computer Graphics and Vision, Graz University of Technology, 2008.

[55] Weiguang Guan. Visualisation of reciprocal space. http://www.rhpcs.mcmaster.ca/ guanw/gallery.html, July 2010.

[56] E. B. Gulsoy, J. P. Simmons, and M. De Graef. Application of joint histogram and mutual information to registration and data fusion problems in

serial sectioning microstructure studies. *Scripta Materialia*, 60(6):381–384, 2009. doi: DOI: 10.1016/j.scriptamat.2008.11.004.

[57] Roger N. Gunn, Steve R. Gunn, Federico E. Turkheimer, John A. D. Aston, and Vincent J. Cunningham. Positron emission tomography compartmental models: A basis pursuit strategy for kinetic modeling. *Journal of Cerebral Blood Flow and Metabolism*, 22:635–652, 2002.

[58] Richard W. Hamming. *Coding and Information Theory.* Prentice Hall, second edition, 1986.

[59] Mark Harris. Mapping computational concepts to gpus. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 50, New York, NY, USA, 2005. ACM.

[60] K. K. Herfarth, J. Debus, F. Lohr, M. L. Bahner, P. Fritz, A. Hss, W. Schlegel, and M. F. Wannenmacher. Extracranial stereotactic radiation therapy: set-up accuracy of patients treated for liver metastases. *International Journal of Radiation Oncology*Biology*Physics*, 46(2):329–335, 2000. doi: DOI: 10.1016/S0360-3016(99)00413-7.

[61] Francis S. Hill. *Computer Graphics Using OpenGL.* Pearson Education, 3 edition, 2008.

[62] Robert R. Hines. Rural america and the great depression. http://www.alamo.edu/pac/faculty/rhines/1302Theme8.htm, June 2010.

[63] Anthony K. Ho, Dongshan Fu, Cristian Cotrutz, Steven L. Hancock, Steven D. Chang, Iris C. Gibbs, Jr. Maurer, Calvin R., and Jr. Adler, John R. A study of the accuracy of cyberknife spinal radiosurgery using skeletal structure tracking. *Neurosurgery*, 60(2):147–156, 2007.

[64] Shan Hongzhang and Singh Jaswinder Pal. A comparison of mpi, shmem and cache-coherent shared address space programming models on a tightly-coupled multiprocessors. *Int. J. Parallel Program.*, 29(3):283–318, 2001. 608776.

[65] Melissa Memorial Hospital. Magnetic resonance imaging (mri). http://www.melissamemorial.org/CMS/Show?id=18, 18 September, 2010.

[66] Zhao Huaxia and A.J. Reader. Fast ray-tracing technique to calculate line integral paths in voxel arrays. *IEEE Nuclear Science Symposium*, 4:2808 – 2812, 2003.

[67] Gregory Michael Hunter. *Efficient computation and data structures for graphics.* PhD thesis, Princeton, NJ, USA, 1978.

[68] Daniel C. Hyde. *Introduction to the Programming Language Occam.* Bucknell University, 1995.

[69] Paul Turner IkkJin Ahn, Michael Lehr. Image processing on the GPU. Technical report, University of Pennsylvania, February 27 2005.

[70] Imaginis. Milestones in medical diagnosis and diagnostic imaging. http://www.imaginis.com/faq/milestones-in-medical-diagnosis-and-diagnostic-imaging, 16 September, 2010.

[71] ImPACT. Radiation therapy treatment planning process. http://www.impactscan.org/slides/impactcourse/introduction_to_ct_in_radiotherapy/img3.html, October, 2010.

[72] Apple Inc. Opencl programming guide for mac os x. Technical report, 10-06-2009.

[73] SII NanoTechnology Inc. Xrf analysis. http://www.siint.com/en/products/xrf/tec_descriptions/descriptions_e.html, September, 2010.

[74] Wald Ingo, Slusallek Philipp, Benthin Carsten, and Wagner Markus. Interactive rendering with coherent ray tracing. *Computer Graphics Forum*, 20(3):153–164, 2001. 10.1111/1467-8659.00508.

[75] Krzysztof. Iniewski. *Medical Imaging : Principles, Detectors, and Electronics.* Hoboken : John Wiley and Sons, Inc., 2009., first edition, 2009.

[76] Fumihiko Ino, Jun Gomita, Yasuhiro Kawasaki, and Kenichi Hagihara. A gpgpu approach for accelerating 2-d/3-d rigid registration of medical images. In *Parallel and Distributed Processing and Applications*, pages 939–950. 2006. 10.1007/11946441-84.

[77] International RadioSurgery Association IRSA. Radiation therapy, 2008.

[78] Medwig J., Gaede S., Battista J., and Yartsev S. Effect of lateral target motion on image registration accuracy in ct-guided helical tomotherapy: A phantom study. *Journal of Medical Imaging and Radiation Oncology*, 54(3):280–286, 2010.

[79] Van Dyk J., T. Kron, G. Bauman, and J. Battista. Tomotherapy: A "revolution" in radiation therapy. *Physics in Canada*, 58:79–86, 2002.

[80] C.L. Jackins and S.L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. 14(3):249–270, November 1980.

[81] Fuegi John and Francis Jo. Lovelace & babbage and the creation of the 1843 'notes'. *IEEE Ann. Hist. Comput.*, 25(4):16–26, 2003. 1436041.

[82] RDCS Karen L. Strub, BS. Adult echocardiography and doppler. *Journal of Diagnostic Medical Sonography*, 21(2):91–110, 2005.

[83] Richard A. Ketcham and William D. Carlson. Acquisition, optimization and interpretation of X-ray computed tomographic imagery: applications to the geosciences. *Comput. Geosci.*, 27:381–400, 2001.

[84] Ali Khamene, Peter Bloch, and Wolfgang Wein et.al. Automatic registration of portal images and volumetric ct for patient positioning in radiation therapy. *Medical Image Analysis*, 10:96–112, 2006.

[85] Soo Mee Kim, Jae Sung Lee, Hee Seo, Jin-Hyung Park, Chan Hyeong Kim, Chun Sik Lee, Myung Chul Lee, Soo-Jin Lee, and Dong Soo Lee. Accelerated reconstruction with resolution recovery modeling for compton camera. *J Nucl Med Meeting Abstracts*, 51(2 Meeting Abstracts):1341, 2010.

[86] D. Knaan and L. Joskowicz. Efficient intensity-based 2D/3D rigid registration between fluoroscopic X-ray and CT. In *MICCAI 2003, 6th Int. Conf., Montréal, Canada, Part I, vol 2878 of Lecture Notes in Computer Science*, pages 351–358. Springer, 2003.

[87] Aaron Knoll, Ingo Wald, Steven Parker, and Charles Hansen. Interactive isosurface ray tracing of large octree volumes. In *In Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, pages 115–124, 2006.

[88] Jan Kok. Parallel programming with ada. *International Journal of High Performance Computing Applications December*, 2(4):100–108, December 1988.

[89] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proc. SIGGRAPH '94*, pages 451–458, Orlando, Florida, July 1994.

[90] David A. LaRose. *Iterative X-ray/CT Registration Using Accelerated Volume Rendering*. PhD thesis, Carnegie Mellon University, 2001.

[91] Sherwood Lauralee. *Fundamentals of Physiology: A Human Perspective*. Brooks Cole, 2 edition, 1994.

[92] Thomas M. Lehmann, Claudia Gnner, and Klaus Spitzer. Survey: Interpolation methods in medical image processing. *IEEE Transactions on Medical Imaging*, 18:1049–1075, 1999.

[93] M. Levoy and P. Hanrahan. Light field rendering. *23rd Annu. Conf. Comput. Graph. Interact. Tech. SIGGRAPH*, 96:3142, 1996.

[94] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, pages 249–261, July 1990.

[95] Sukhyun Lim and Byeong-Seok Shin. A distance template for octree traversal in cpu-based volume ray casting. *The Visual Computer*, 24:229–237, 2008. 10.1007/s00371-007-0203-y.

[96] Jerry L. Prince Links and Jonathan M. *Medical Imaging Signals and Systems*. Pearson Prentice Hall, first edition, 2006.

[97] C. Liu, R. Schmitz, K. Hendrickson, S. Patel, R. Jeraj, and P. Kinahan. Su-gg-j-142: Investigation of mip pet in motion-encompassing methods to account for respiratory motion in radiotherapy. volume 37, pages 3178–3178. AAPM, 2010.

[98] Yuqiang Lu, Weiming Wang, Shifu Chen, Yongming Xie, Jing Qin, Wai-Man Pang, and Pheng-Ann Heng. Accelerating algebraic reconstruction using cuda-enabled gpu. *International Conference on Computer Graphics, Imaging and Visualization*, 0:480–485, 2009.

[99] Ollinger J. M. and J. A. Fessler. Positron-emission tomography. *IEEE Signal Processing Magazine*, 14:43–55, Jan 1997.

[100] Frederik Maes, Andr Collignon, Dirk V, Guy Marchal, and Paul Suetens. Multimodality image registration by maximization of mutual information. *IEEE transactions on Medical Imaging*, 16:187–198, 1997.

[101] Paprzycki Marcin and Zalewski Janusz. Parallel computing in ada: an overview and critique. *Ada Lett.*, XVII(2):55–62, 1997. 249112.

[102] Marketwire. Tomotherapy enters australian market with purchase by royal brisbane and women's hospital. http://www.marketwire.com/press-release/TomoTherapy-Enters-Australian-Market-With-Purchase-by-Royal-Brisbane-Womens-Hospital-NASDAQ-TOMO-1114371.htm, September, 2010.

[103] S. Mashohor, J.R. Evans, and A.T. Erdogan. Image registration of printed circuit boards using hybrid genetic algorithm, 2006.

[104] Georges Palos Jean-Yves Gauvrit Christian Vasseur Maximilien Vermandel, Nacim Betrouni and Jean Rousseau. Registration, matching, and data fusion in 2d/3d medical imaging: Application to dsa and mra. *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003*, 2878:778–785, 2003.

[105] J. McDonough, I. Rusu, L. Wang, M. Dimcovski, Z. Dimcovski, and P. Bloch. Dosimetric evaluation of an amorphous silicon portal imaging device. *International Workshop on Electronic Portal Imaging*, page 130132, 2002.

[106] Babu M. Mehtre, Mohan S. Kankanhalli, and Lee Wing Foon. Shape measures for content based image retrieval: A comparison. *Information Processing and Management*, 33(3):319–337, 1997. doi: DOI: 10.1016/S0306-4573(96)00069-6.

[107] C.T. Metz. Digitally reconstructed radiographs. Master's thesis, Utrecht University, 2005.

[108] Prism Microsystem. The many sides of prism. http://www.prism.co.uk/, October, 2010.

[109] N. Milickovic, D. Baltas, S. Giannouli, M. Lahanas, and N. Zamboglou. Ct imaging based digitally reconstructed radiographs and their application in brachytherapy. *Physics in Medicine and Biology*, 45:2787–2800, 2000.

[110] B. Amin Minesh, Grama Ananth, and Singh Vineet. Fast volume rendering using an efficient, scalable parallel formulation of the shear-warp algorithm, 1995. 2183307-14.

[111] Shinichiro Mori, Masanao Kobayashi, Motoki Kumagai, and Shinichi Minohara. Development of a gpu-based multithreaded software application to calculate digitally reconstructed radiographs for radiotherapy. *Radiological Physics and Technology*, 2(1):40–45, 2009. 10.1007/s12194-008-0040-3.

[112] Martin J. Murphy, Steven D. Chang, Iris C. Gibbs, Quynh-Thu Le, Jenny Hai, Daniel Kim, David P. Martin, and John R. Adler. Patterns of patient movement during frameless image-guided radiosurgery. *International Journal of Radiation Oncology*Biology*Physics*, 55(5):1400–1408, 2003. doi: DOI: 10.1016/S0360-3016(02)04597-2.

[113] Katie L. Newbold, Mike Partridge, Gary Cook, Bhupinder Sharma, Peter Rhys-Evans, Kevin J. Harrington, and Christopher M. Nutting. Evaluation

of the role of pet/ct in radiotherapy target definition in patients with head and neck cancer. *Acta Oncologica*, 47(7):1229 – 1236, 2008.

[114] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *Queue*, 6(2):40–53, 2008.

[115] Robert A. Novelline. *Squire's Fundamentals of Radiology*. Harvard University Press, fifth edition, 1997.

[116] NVIDIA. Cg toolkit a developer's guide to programmable graphics. Technical report, September 2005.

[117] NVIDIA. Cuda developer guide for nvidia optimus platforms. Technical report, February 2010.

[118] NVIDIA. Cuda programming guide. Technical report, February 2010.

[119] M.Fisher O.Dorgham. Performance of 2D/3D medical image registration using compressed volumetric data, 2nd-3rd July 2008. 2008.

[120] Canadian Insitutes of Health Research. Research profile - a medical imaging timeline. http://www.cihr-irsc.gc.ca/e/42173.html, 30 July, 2010.

[121] Radiological Society of North America Inc. Linear accelerator. http://www.radiologyinfo.org/en/info.cfm?pg=linac, September, 2010.

[122] Tinsu Pan, Osama Mawlawi, Sadek A. Nehmeh, Yusuf E. Erdi, Dershan Luo, Hui H. Liu, Richard Castillo, Radhe Mohan, Zhongxing Liao, and H. A. Macapinlac. Attenuation correction of pet images with respiration-averaged ct images in pet/ct. *J Nucl Med*, 46(9):1481–1487, 2005.

[123] Graeme Penney, Jrgen Weese, John Little, Paul Desmedt, Derek Hill, and David Hawkes. A comparison of similarity measures for use in 2d-3d medical image registration. In *Medical Image Computing and Computer-Assisted Interventation MICCAI98*, pages 1153–1161. 1998. 10.1007/BFb0056305.

[124] Graeme P. Penney, Philipp G. Batchelor, Derek L. G. Hill, David J. Hawkes, and Juergen Weese. Validation of a two- to three-dimensional registration algorithm for aligning preoperative CT images and intraoperative fluoroscopy images. *Medical Physics*, 28(6):1024–1032, 2001.

[125] Lacroute Philippe and Levoy Marc. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 451–458, New York, NY, USA, 1994. ACM.

[126] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever. Mutual information matching in multiresolution contexts. *Image and Vision Computing*, 19(1-2):45–52, 2001. doi: DOI: 10.1016/S0262-8856(00)00054-8.

[127] Victor R. Preedy and Timothy J. Peters. *Skeletal Muscle: Pathology, Diagnosis and Management of Disease*. Cambridge University Press, UK, 2002.

[128] Wenyuan Qi and Lixu Gu. Effective 2D-3D medical image registration using support vector machine. In *Proc. Annual Int. Conf. of IEEE Engineering in Medicine and Biology Society*, pages 5386–5389, 2008.

[129] Lele R. *Computers In Medicine: Progress In Medical Informatics*. Tata Mcgraw Hill, 2005.

[130] V. R. Kini S. S. Vedam J. V. Siebers Q. Wu R. George, P. J. Keall, D. W. Arthur M. H. Lauterbach, and R. Mohan. Quantifying the effect of intrafraction motion during breast imrt planning and dose delivery. *Medical Physics*, 30(4):552–562, 2003.

[131] Inc. (RSNA) Radiological Society of North America. Positron emission tomography computed tomography (pet/ct). http://www.radiologyinfo.org/en/info.cfm?pg=PET, November, 2010.

[132] Chester R. Ramsey, Don Arwood, Daniel Scaperoth, and Adrian L. Oliver. Clinical application of digitally-reconstructed radiographs generated from

magnetic resonance imaging for intracranial lesions. *International Journal of Radiation Oncology*Biology*Physics*, 45(3):797–802, 1999. doi: DOI: 10.1016/S0360-3016(99)00173-X.

[133] Rost Randi J., Licea-Kane Bill, Ginsburg Dan, Kessenich John M., Malan Barthold, Lichtenbeltand Hugh, and Weiblen Mike. *OpenGL Shading Language*. Addison-Wesley Professional, third edition, 2009.

[134] R. Reddy and S. Rubin. Representation of three-dimensional objects. In *CMU-CS-TR*, 1978.

[135] Liu Ren, Hanspeter Pfister, and Matthias Zwicker. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Computer Graphics Forum (Eurographics 2002)*, volume 21, pages 461–470, September 2002.

[136] Chandra Robit, Dagum Leonardo, Kohr Dave, Maydan Dror, McDonald Jeff, and Menon Ramesh. *Parallel programming in OpenMP*. Morgan Kaufmann Publishers Inc., 2001. 355074.

[137] T. Rohlfing, J. Denzler, D.B. Russakoff, and C.R. Maurer. Markerless realtime 3-d target region tracking by motion backprojection from projection images. 24(11):1455–1468, November 2005.

[138] Philip Ronan. Em spectrum. http://en.wikipedia.org/wiki/File:EM_ spectrum.svg, September, 2010.

[139] W.C. Röntgen. On a New Kind of Rays. *Nature*, 53:274–276, 1896.

[140] Daniel Ruijters, Bart M. ter Haar Romeny, and Paul Suetens. Gpu-accelerated digitally reconstructed radiographs. In *Proc. BioMed2008*, Innsbruck, Austria, February 13–15 2008.

[141] Daniel B. Russakoff, Torsten Rohlfing, and Kensaku Mori et.al. Fast generation of digitally reconstructed radiographs using attenuation fields with application to 2D-3D image registration. *IEEE Transaction on Medical Imaging*, 24:1441–1454, 2005.

[142] Daniel Rypl. Octree data structure. http://power2.fsv.cvut.cz/d̃r/papers/Thesis/node26.html#octree_hierarchy, July 2010.

[143] Miguel Sainz and Renato Pajarola. Point-based rendering techniques. *Computers and Graphics*, 28:869–879, 2004.

[144] Hanan Samet. *The design and analysis of spatial data structures.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[145] David Sarrut and Sbastien Clippe. Fast DRR generation for intensity-based 2D/3D image registration in radiotherapy. Technical Report RR-LIRIS-2003-002, LIRIS UMR 5205 CNRS/INSA de Lyon/Universit Claude Bernard Lyon 1/Universit Lumire Lyon 2/Ecole Centrale de Lyon, December 2003.

[146] David Sarrut and Serge Miguet. Similarity measures for image registration. In *In First European Workshop on Content-Based Multimedia Indexing*, pages 263–270, 1999.

[147] Sohail Sayeh, James Wang, William Main, Warren Kilby, and Calvin Maurer. Respiratory motion tracking for robotic radiosurgery. In *Treating Tumors that Move with Respiration*, pages 15–29. Springer Berlin Heidelberg, 2007.

[148] Lisa B. Schwartz, David L. Olive, and Shirley McCarthy. *Diagnostic Imaging for Reproductive Failure.* Taylor and Francis Ltd, UK, 1998.

[149] We Care Health Services. Cancer treatment (radiotherapy). http://www.wecareindia.com/cancer-treatment/radiotherapy-treatment.html, October, 2010.

[150] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, 2001.

[151] Hao-Chieh Chang Sheng-Chieh Huang, Liang-Gee Chen. A novel image compression algorithm by using log-exp transform. In *IEEE International Symposium on Circuits and Systems*, volume 4, pages 17–20, 1999.

[152] Lawrence A. Shepp. *Computed Tomography*, volume 27. American Mathematical Society, 1983.

[153] G.W. Sherouse, K. Novins, B.S. Chaney, and E.L. Chaney. Computation of digitally reconstructed radiographs for use in radiotherapy treatment design. *Int. J. Radiation Oncology Biol. Phys.*, 18(3):651–658, March 1990.

[154] K. Sikora, S. Advani, V. Koroltchouk, I. Magrath, L. Levy, H. Pinedo, G. Schwartsmann, M. Tattersall, and S. Yan. Essential drugs for cancer therapy: Aworld health organization consultation. *Annals of Oncology*, 10:385–390, 1999.

[155] Blair Silver and Karen Panetta. Contrast entropy based image enhancement and logarithmic transform coefficient histogram shifting. *ICASSP*, 2005.

[156] Brian Smits. Efficiency issues for ray tracing. *Journal of Graphics Tools*, 3:1–14, 1999.

[157] Weiwei Song, Shungang Hua, Zongying Ou, Hu An, and Kaifeng Song. Octree based representation and volume rendering of three-dimensional medical data sets. *BioMedical Engineering and Informatics, International Conference on*, 1:316–320, 2008.

[158] Milan Sonka and J. Michael Fitzpatrick, editors. *Handbook of Medical Imaging, Volume 2. Medical Image Processing and Analysis*. SPIE Press, Bellingham, WA, 2000.

[159] Grimm Soren, Bruckner Stefan, Kanitsar Armin, and Groller Eduard. Memory efficient acceleration structures and techniques for cpu-based volume raycasting of large data, 2004. 10390171-8.

[160] Jakob Spork. High-performance gpu based rendering for real-time, rigid 2d/3d-image registration in radiation oncology, 2010.

[161] Perry Sprawls. The web-based edition of the physical principles of medical imaging. http://www.sprawls.org/ppmi2/, August, 2010.

[162] F. Zeigler Stephen and P. Weicker Reinhold. Ada language statistics for the imax 432 operating system. *Ada Lett.*, II(6):63–67, 1983. 989818.

[163] Y. Su, M. Fisher, and R. Rowland. Marker-less intra-fraction organ motion tracking using hybrid asm. *International Journal of Computer Assisted Radiology and Surgery*, 2:231–243, 2007. 10.1007/s11548-007-0133-1.

[164] VARIAN Medical Systems. Clinac linear accelerator. http://varian.mediaroom.com/index.php?s=13&cat=12&mode=gallery, September, 2010.

[165] VARIAN Medical Systems. Smartbeam imrt better technology at every step. http://www.varian.com/eude/oncology/treatments/treatment_techniques/IMRT/, September, 2010.

[166] M. Tamminen. Comment on quad- and octtrees. *Communications of the ACM*, 27(3):248–249, March 1984.

[167] Bin S. Teh, Shiao Y. Woo, and E. Brian Butler. Intensity modulated radiation therapy (imrt): A new promising technology in radiation oncology. *Oncologist*, 4(6):433–442, 1999.

[168] Gross Tobias. 2D/3D registration of X-Ray and CT data using accelerated volume rendering and gradient calculation. *11th Central European Seminar on Computer Graphics*, 2007.

[169] CT TOMS RIVER X-RAY and MRI CENTER. CT (cat) scan. http://www.tomsriverxray.com/ct.nxgv, October, 2010.

[170] Gerard J. Tortora, Nicholas P. Anagnostakos, and Sandra Reynolds Grabowski. *Principles of Anatomy and Physiology.* John Wiley and Sons Inc, 7th edition, 1993.

[171] A. Trn and A. Zilinskas. *Global Optimisation.* Springer-Verlag, Berlin, 1989.

[172] Kumar Vipin, Grama Ananth, Gupta Anshul, and Karypis George. *Introduction to parallel computing: design and analysis of algorithms.* Benjamin-Cummings Publishing Co., Inc., 1994.

[173] Torczon Virginia. On the convergence of pattern search algorithms. *SIAM J. on Optimization*, 7(1):1–25, 1997. 589066.

[174] Michael Vogele and Andrew Long. Patent application title: Radiotherapeutic apparatus. http://www.faqs.org/patents/app/20090238338, September, 2010.

[175] Gustav von Schulthess. Integrated modality imaging with PET-CT and SPECT-CT: CT issues. *European Radiology Supplements*, 15(0):d121–d126, 2005.

[176] G. Sakas W. Cai. Drr volume rendering using splatting in shear-warp context. *Nuclear Science Symposium Conference Record, 2000 IEEE*, 3, 2000.

[177] Xuzhu Wang, B. De Baets, and E. Kerre. A comparative study of similarity measures. *Fuzzy Sets and Systems*, 73(2):259–268, 1995. doi: DOI: 10.1016/0165-0114(94)00308-T.

[178] Z. L. Wang et al. Dynamic linear level octree-based volume rendering methods for interactive microsurgical simulation. *Int. Journal of Image and Graphics*, 6(2):155–171, April 2006.

[179] Andrew G. Webb. *Introduction to Biomedical Imaging.* Wiley, John and Sons, Incorporated, first edition, 2003.

[180] S. Webb. *Contemporary IMRT Developing Physics and Clinical Implementation.* Taylor and Francis, first edition, 2004.

[181] Juergen Weese, Roland Goecke, Graeme P. Penney, Paul Desmedt, Thorsten M. Buzug, and Heidrun Schumann. Fast voxel-based 2D/3D registration algorithm using a volume rendering method based on the shear-warp factorization. volume 3661, pages 802–810. SPIE, 1999. Medical Imaging 1999: Image Processing 1.

[182] Wolfgang Wein. *Intensity Based Rigid 2D-3D Registration Algorithms for Radiation Therapy.* PhD thesis, TU Munich, 2003.

[183] Song Weiwei, Hua Shungang, Ou Zongying, An Hu, and Song Kaifeng. Octree based representation and volume rendering of three-dimensional medical data sets, 2008. 1372181 316-320.

[184] M. Wieczorek, A. Aichert, O. Kutter, C. Bichlmeier, J. Landes, S. M. Heining, E. Euler, and N. Navab. GPU-accelerated Rendering for Medical Augmented Reality in Minimally-Invasive Procedures. In *Proceedings of BVM 2010*. Springer, Mar. 2010.

[185] Anthony Brinton Wolbarst. *Looking Within: How X-Ray, CT, MRI, Ultrasound, and Other Medical Images Are Created, and How They Help Physicians Save Lives*. University of California Press, 1999.

[186] Birkfellner Wolfgang, Seemann Rudolf, Figl Michael, Hummel Johann, Ede Christopher, Homolka Peter, Xinhui Yang, Peter Niederer, and Helmar Bergmann. Wobbled splatting a fast perspective volume rendering method for simulation of X-ray images from CT. *Physics in Medicine and Biology*, 50(9):N73–N84, 2005.

[187] Ziv Yaniv and Kevin Clearly. Image guided procedures: A review. Technical Report TR-2006-3, Georgetown University, Image Science and Information Centre, Washington DC, April 2006.

[188] S Yartsev, T Kron, and J Van Dyk. Tomotherapy as a tool in image-guided radiation therapy (igrt): current clinical experience and outcomes. *Biomed Imaging Interv J*, 3(1):e17, 2007.

[189] S Yartsev, T Kron, and J Van Dyk. Tomotherapy as a tool in image-guided radiation therapy (igrt): theoretical and technological aspects. *Biomed Imaging Interv J*, 3(1):e16, 2007.

[190] M.M. Yau and S.N. Srihari. A hierarchical data structure for multidimensional digital images. *Communications of the ACM*, 26(7):504–515, July 1983.

[191] Wu Yung-Gi. Region of interest image indexing system by dct and entropy. *GVIP Journal*, Volume 6(Issue 4):10, 2006.