FAST AND ACCURATE THREE-DIMENSIONAL RECONSTRUCTION FROM CONE-BEAM PROJECTION DATA USING ALGEBRAIC METHODS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for

the Degree Doctor of Philosophy in the Graduate

School of The Ohio State University

By

Klaus Mueller, Dipl. Ing. (FH), M.S., M.S.

* * * * *

The Ohio State University
1998

Dissertation Committee:

Dr. Roni Yagel, Adviser

Dr. Roger Crawfis

Dr. Richard Parent

Approved by

_____

Adviser

Department of Computer and Information Science

ABSTRACT

Cone-beam computed tomography (CT) is an emerging imaging technology, as it provides all projections needed for three-dimensional (3D) reconstruction in a single spin of the X-ray source-detector pair. This facilitates fast, low-dose data acquisition as required for imaging fast moving objects, such as the heart, and intra-operative CT applications. Current cone-beam reconstruction algorithms mainly employ the Filtered-Backprojection (FBP) approach. In this dissertation, a different class of reconstruction algorithms is studied: the algebraic reconstruction methods. Algebraic reconstruction starts from an initial guess for the reconstructed object and then performs a sequence of iterative grid projections and correction backprojections until the reconstruction has converged. Algebraic methods have many advantages over FBP, such as better noise tolerance and better handling of sparse and non-uniformly distributed projection datasets. So far, the main repellant for using algebraic methods in routine clinical situations was their slow speed. Apart from providing solutions for this pressing problem, we will also apply, for the first time, algebraic methods in the context of general low-contrast cone-beam tomography. This new context poses several challenges, both for reconstruction quality and speed.

To eliminate the strong aliasing effects that occur when standard algebraic methods are applied for cone angles exceeding 20˚, we introduce the new concept of depth-adaptive basis function kernels. Then, a comprehensive study is conducted on how various parameters, such as grid initialization, relaxation coefficient, number of iterations, and correction method (ART or SART) influence cone-beam reconstruction quality and speed. Finally, a new algorithm, the Weighted Distance Scheme, is proposed that optimally arranges the order in which the grid projections are performed., which reduces the number of iterations and promotes reconstruction quality. We find that three iterations are sufficient to obtain good reconstruction quality in the general case. A cost function is then developed that relates the computational effort of FBP with that of algebraic methods. In an attempt to match the cost of algebraic methods and FBP, a new accurate and efficient projection algorithm is proposed that reaches its goal by caching projection computations for reuse in the subsequent backprojection. Finally, we propose a new hardware-accelerated scheme for algebraic methods that utilizes readily available off-the-shelf graphics hardware and enables a reconstruction to be performed in less than 2 minutes at good quality.

Gewidmet meiner Mutter und meinem Vater

ACKNOWLEDGMENTS

If you spend ten years of your life at one place, like I did at Ohio State, you will get to know a wide variety of people that are affiliated with this institution in one way or the other. Many of these people will be instrumental in the success of your work, and without their help and inspiration you will never reach your goals, or even be able to discover and define them. My path since arriving from Germany with two suitcases in my hands to the present load, filling a medium sized U-Haul truck, was lined with some of the nicest and most resourceful people I have ever met in my life. I was able to build many incredible friendships and collaborations, and I fondly look back to my time as a graduate student at Ohio State.

The person who, without a doubt, has inspired and supported me the most is my advisor Roni Yagel, a mentor *par excellence*. A meeting with him was like a complete mental overhaul: I would walk into his office, deeply worried about some matter, and would leave an hour later, re-vitalized and full of hope, eager to attack the problem, whatever it was. There was never a time where he could not spare a moment - and often, a moment turned into an hour and more. His optimism, enthusiasm, keen sense for opportunity, ingenuity, and friendship have taught me a great deal and have helped me to grow into a person much more mature than I was when I first met him in 1992.

Another person I am deeply indebted to is Ed Herderick of the Biomedical Engineering (BME) Center at Ohio State, whom I have known almost since my beginnings at this university. He was the one who recommended me for my first assistantship, inspired, supervised, and guided me through my Master's thesis at BME, and provided me with support ever since, even after my departure from BME. The road of this dissertation would have been a lot rockier without his permission to maintain my "home" at the Laboratory of Vascular Diseases, whose splendid and neatly maintained computational resources and comfortable premises have made it so much easier to accomplish this research. In this respect, I also thank Dr. Mort Friedman, Acting Chairman of BME, for generously letting me heat the BME CPU's and spin 3 GB of harddrive real-estate over the past two years. Thanks are especially in order for BME system administrator Vlad Marukhlenko who kept the CPU's heatable, the harddrives spinable, and the cooling fans quiet.

Enthusiastic thank-you's go out to a number of people that have provided me with research opportunities and financial backing, and who have trusted in me that I could actually perform the work that had to be done. First, I must thank Prof. Herman Weed, and also Dr. Richard Campbell, who admitted me to the BME program in a rather un-bureaucratic manner, not knowing much about me other than my grades and the fact that I was German. When I arrived in Columbus one cold winter day in 1988, the wind hissing around the cor-

ners of Dreese Laboratories, Prof. Weed and Dr. Campbell quickly integrated me into the program, and made sure I wouldn't get lost so far away from home.

I am also greatly indebted to Dr. J. Fredrick Cornhill, who became my advisor shortly after my arrival in the US and who would let me do computer-related research even though I, as an electrical engineer, was initially not very versed in these aspects. This early research sparked my interest in computer science, and Dr. Cornhill's generous funding for the next seven years enabled me to deeply acquaint myself with the subject, a task that ultimately lead to a second Master's degree in that discipline (and subsequently this Ph.D.). I want to thank Dr. Cornhill for providing me with the opportunity to participate in a number of clinical software projects at the Cleveland Clinic Foundation. This involvement in real-life applications of computers in the medical field have surely prepared me well for a career that is geared towards the development of practical systems that benefit human health and well-being.

I would also like to thank Dr. John Wheller, head of the Columbus Children's Hospital catheterization lab, whose innovative effort for better medical imaging technologies has provided the clinical motivation of this dissertation research. I sincerely appreciate the time he has invested in writing the grant that has supported me for two years, his permission to letting me access the medical facilities at the hospital, his eagerness to understand the technical aspects of the project, and his brave struggle with bureaucracy to keep things rolling.

Last but not least, I shall thank Dr. Roger Crawfis, who joined Ohio State just a year ago, and who supported me for the past six months. I have had a great final lap at OSU, thanks to Roger, with some rather interesting research projects popping up in the past year, opening many promising prospects for the years ahead.

I shall continue the list with thanking the adjunct members of my dissertation committee, Dr. Rick Parent and Dr. Roger Crawfis for reading the document, enduring the 113 slides in my oral defense, and providing many helpful comments.

Another marvelous individual that I had the pleasure to meet and collaborate with is Dr. Robert Hamlin, Professor at the Department of Veterinary Physiology. A sign posted at his door says: "Don't knock, just walk in!", and that's exactly how he works: Whenever I needed something, be it a letter of recommendation or a person to serve on a committee, Dr. Hamlin was right there for me. Thanks for your lively instructions on the cardiac and circulatory system, and the research work we have done together with Randolph Seidler.

Corporate support over the years was generously granted by both Shiley-Pfizer and General Electric. In conjunction with the latter, I'd like to thank both Steve Chucta, technician at the Children's Hospital catheterization lab and GE field engineer Mike Brenkus for their technical assistance with the GE Angiogram Scanner. Without Mike's midnight job on wiring-up the scanner boards we would have never been able to spatially register the angiograms for 3D reconstruction. With respect to the Pfizer project, I'd like to acknowledge Dr. Kim Powell and Eric LaPresto from the Cleveland Clinic Biomedical Engineering Department, and Jamie Lee Hirsch and Dr. James Chandler from the Shiley Heart Valve Research Center. It was a great experience to work with this team, developing algorithms and soft-

ware systems to diagnose defective Björk-Shiley artificial heart valves in-vivo. Thanks especially to Kim Powell for two years of fun collaboration, co-authorship and the windy weekends on 'Breakaway' with Craig, and Eric LaPresto for a great job on smoothing out our final paper.

This list is definitely not complete without mentioning the M^3Y-factory, born out of a casual Biergarten project, and matured into a serious (well...) research enterprise, producing so far three conference papers and one journal paper on exploring new ways for calculating more accurate interpolation and gradient filters for volume rendering. The M^3Y partners in crime are: Torsten Möller, Raghu Machiraju, and Roni Yagel. Thanks guys, for your continued friendship and the opportunity of racing each of the papers 10 minutes before deadline at 95 mph to the nearest DHL drop-off station.

Thanks also to the "Splatters": Ed Swan, Torsten Möller, Roger Crawfis, Naeem Shareef, and Roni Yagel, which formed a collaboration with the aim of producing more accurate perspective volume renderings with the splatting algorithm. The ideas that emerged in this project led to the depth-adaptive kernels that were also used in this dissertation. I would like to thank in particular Ed Swan and Torsten Möller for passing the honor of first authorship in the journal paper on to me. Turning the paper around from its depressing first review to the completely overhauled final version was just a great accomplishment by all participants. The nightly sessions with math-wiz Torsten were a special treat.

My thanks also go out to Roger Johnson, formerly Professor at BME, for enlightening me both on the physics part of medical imaging and the bourbon distillation process, and for meticulously proof-reading an early journal paper on my research. It was nice to have him as an inofficial member in my dissertation committee.

Don Stredney of the Ohio Supercomputer Center (OSC) deserves a big thank-you for letting me use OSC's large-caliber computing equipment for developing my interactive volume renderer and refining the texture-mapping hardware accelerated ART algorithm. Don was also the one who brought me in contact with Dr. Wheller, and who helped me out with obtaining clinical volume data sets that I could dig my volume renderer's teeth in. Likewise, Dennis Sessanna is thanked for his assistance at the OSC computing facilities.

The texture-mapping hardware accelerated ART algorithm was developed during a two-month internship at Silicon Graphics Biomedical in Jerusalem, Israel. I really enjoyed my time there, thanks to the great hospitality that was extended to me by the people working at the company, in particular: Michael Berman, Ziv Sofermann, Lana Tocus, Dayana, and my host Roni Yagel.

Finally, I should thank all my friends that made sure that I would not spend *all* my time in the lab and who dragged me out into the fresh air every once in a while (listed in random order, without claiming completeness): Perry Chappano, Derek Kamper, Anne Florentine, Field Hockey All-American Britta Eikhoff, Michael Malone, die Skat Brüder Norbert Peekhaus, Ulf Hartwig, und Glenn Hofmann, Anton (and Claire) Bartolo, whose unrelated presence in the lab at night gave me the assuring confidence that I wasn't the only one trading beers for work, Enrico Nunziata, Duran and Mary-Jane Yetkinler, Randolph Seidler

who bought me my first Schweinshaxn, Glenn and Monique Nieschwitz, Holger “Don’t worry - be Holgi” Zwingmann, the Martyniuk-Pancziuk’s, Randall and Ivana Swartz, Matt and Vanessa Waliszewski, VolViz comrades Torsten Möller, Naeem Shareef, Raghu Machiraju, Ed Swan, Yair Kurzion and Asish Law, Chao-hui Wang, Linda Ludwig who provided me with a bending shelf of free Prentice-Hall text books, Madhu Soundararajan, the BME gang Ajeet Gaddipati, Bernhard (and Christina) Sturm, the Kimodos Bala Gopakumaran and Andre van der Kouwe,...

But, most importantly, I must thank my parents, Rolf and Gisela Müller, and my sister Bärbel, for never giving up on me and always supporting me, no matter how odd my next move seemed to be. This dissertation shall be dedicated to you.

VITA

October 12, 1960. . . . . Born - Stuttgart, Germany

1987. . . . . Dipl. Ing. (FH), Polytechnic University Ulm, Germany

1990. . . . . M.S. (BME), The Ohio State University

1996. . . . . M.S. (CIS), The Ohio State University

1988 - present . . . . . Graduate Research Associate, The Ohio State University

PUBLICATIONS

Publications relevant to the topic of this dissertation:

[1] K. Mueller, R. Yagel, and J.J. Wheller, “A fast and accurate projection algorithm for 3D cone-beam reconstruction with the Algebraic Reconstruction Technique (ART),” *Proceedings of the 1998 SPIE Medical Imaging Conference*, Vol. SPIE 3336, pp. (in print), 1998. (won Honorary Mention Award.)

[2] K. Mueller, R. Yagel, and J.F. Cornhill, “The weighted distance scheme: a globally optimizing projection ordering method for the Algebraic Reconstruction Technique (ART),” *IEEE Transactions on Medical Imaging*, vol. 16, no. 2, pp. 223-230, April 1997.

[3] K. Mueller, R. Yagel, and J.F. Cornhill, “Accelerating the anti-aliased Algebraic Reconstruction Technique (ART) by table-based voxel backward projection,” *Proceedings EMBS’95 (The Annual International Conference of the IEEE Engineering in Medicine and Biology Society)*, pp. 579-580, 1995.

FIELDS OF STUDY

Major Field: Computer and Information Science (Computer Graphics)

Minor Field: Artificial Intelligence

Minor Field: Statistics

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

There is no doubt that G.N. Hounsfield's invention of the Computed Tomography (CT) scanner in 1972 has revolutionized diagnostic medicine. This invention has not only gained Hounsfield the honors of British Knighthood and the Nobel Prize in Medicine in 1979, but has also brought millions of dollars to the manufacturing company EMI, Ltd. and to others that have licensed the patent. But most importantly, it has, for the first time, made it possible to visually inspect a person's inside anatomy without the need for invasive surgery. Hounsfield's invention showed that it was principally feasible, based on a very large number of measurements, to reconstruct a cross-sectional slice of a patient with fairly high accuracy. In the following years, the image quality of the slices improved drastically. Two main reconstruction techniques emerged, which still exist up to this day: On one side there is the domain of direct methods that capitalize on the Fourier Slice Theorem [4], while on the other side lies the domain of iterative methods that seek to solve the reconstruction problem by solving a system of simultaneous linear equations. The most prominent member of the former group is the Filtered Backprojection (FBP) algorithm [5][50]. Here, the reconstruction is achieved by filtering the projection images with a ramp filter in frequency space, and then backprojecting the filtered projections onto a reconstruction grid. The first and still most prevalent representative of the iterative methods is the Algebraic Reconstruction Technique (ART), attributed to Gordon et. al. [17]. Here, an object is reconstructed on a discrete grid by a sequence of alternating grid projections and correction backprojections. The projection measures how close the current state of the reconstructed object matches one of the scanner projections, while in the backprojection step a corrective factor is distributed back onto the grid. Many such projection/backprojection operations are typically needed to make the reconstructed object fit all projections in the acquired set, within a certain tolerance margin.

While two-dimensional (2D) slice-based CT has been in clinical use for many years, the current trend is now to three-dimensional (3D) or volumetric CT, in which projection acquisition is directly geared towards a 3D reconstruction of the object. This is in contrast to 3D representations obtained with standard slice-based CT, in which a number of sepa-

rately acquired, thin axial slices are simply stacked on top of one another.

One of the remaining great challenges in CT (and also other imaging modalities, like Magnetic Spin Resonance (MRI)) is the 3D reconstruction of moving objects, such as the beating heart, along with the coronary vessels and valves, and the lungs. Although these organs move in a cyclic fashion, they do so in slightly varying trajectories. Respiratory motion of the patient also contributes to this spatial variation. By gating image acquisition with a physiological signal, such as the ECG, one can tag the acquired images and associate them with a particular time instance within the organ's cycle. In this way, images of equal time instances can be collected into a set and used for the 3D reconstruction of the object at that particular time frame. One can so generate animated, four-dimensional (4D) reconstructions of the object, with the fourth dimension being time. However, the irregularity of the cycle trajectory causes inconsistencies within the images of a time set, since these images have not really been acquired at the *same* time instance, but only at *equivalent* time instances, a few cycles apart. This registration error will manifest itself in the reconstructions as blurring and streaking.

The ultimate device to conduct this kind of dynamic imaging is the Dynamic Spatial Reconstructor (DSR), built as early as 1983 at the Mayo Clinic by Richard Robb [53][54] (shown in Figure 1.1). Here, 14 rotating 2D cameras with 240 scan lines each receive photons of 14 opposing X-ray point sources at a frequency of 1/60 seconds. Thus, if a reconstruction algorithm requires, say 42 images, for faithful reconstruction, and the imaged organ does not move much within this window of 4/60 seconds, then we can reconstruct this organ without the multi-cycle motion artifacts mentioned above, at a temporal resolution of 4/60 seconds. There are, however, a few problems with the DSR: First, the scanner is very expensive, due to the replicated X-ray tubes and detectors, the heavy machinery required to rotate the gantry, and the electronics required to control image acquisition. Second, the gantry rotates only 1.5˚ per 1/60 sec, which means that the images per time window are not uniformly distributed in orientation angle. Hence, gated imaging is still necessary, although over a much reduced number of heart cycles.

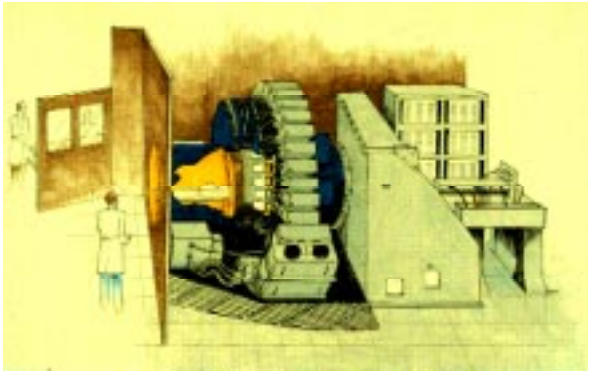


FIGURE 1.1: Dynamic Spatial Reconstructor (DSR) built in 1983 at Mayo Clinic, Rochester, MN. (Image was obtained from http://everest.radiology.uiowa.edu/gallery/dsr.html).

At the time the DSR was developed, 3D reconstruction was still in its infancy and no true 3D reconstruction algorithm was available. In need of pioneering results, the DSR was forced to employ a standard 2D reconstruction algorithm, originally designed to reconstruct cross-sectional slices from fan-beam projection data. In an approach termed the stack-of-fans method, the DSR simply treated each axial row of projection data as coming from a rotating virtual 2D fan-beam source, located in the same plane. This approximation yielded reasonable results, which was not surprising since the source-detector distance (i.e., the gantry diameter) was quite large, reducing the subtended angle of the X-ray cone to a mere 4°. Hence, the fans in the stack-of-fans were nearly parallel. A modern 3D scanner, however, would employ much larger cone angles for the following reasons:

- smaller scanner design,
- to increase the utilization of the radially emitted X-rays,
- better magnification,
- to expand the reach in the axial direction, and
- to accommodate larger detector arrays.

As a matter of fact, micro-tomography (e.g., used to investigate the progress of osteoporosis in trabecular bone) uses cone-angles of up to 60°. It should also be noted at this point that the DSR has never been duplicated, and rumor has it that it is currently being dismantled for scrap.

If dynamic imaging is to be performed in a widespread clinical setting, it is probably more economical to resort to imaging equipment that is readily available at hospitals, such as a biplane C-arm X-ray scanner [11] (shown in Figure 1.2) or a standard CT-gantry with an added 2D camera [47][48][60] (similar to the DSR, just with one camera instead of 14). While the latter still requires modification of standard hospital equipment, however, at a smaller cost than the DSR, the former is available ready-to-go in great abundance.

Just to illustrate the efficacy of the former solution, let me give an example: The gantry of a standard biplane C-arm scanner rotates at maximal speeds of 10°/s around the patient and acquires images at 30 frames/s per camera. With the biplane setup, we only need a little more than one quarter spin to cover the necessary angular range of about 220°. Thus, with an acquisition time of 11s, we obtain 660 images (with the two cameras). If we were to image a patients heart, perfused with contrast dye, we would cover 11 patient heart beats (assuming the heart rate is the standard 1bps) with this amount of image data. If we then divided a heart beat into 15 time slots (like the DSR), we would have 44 images per time slot, fairly uniformly distributed over the semi-circle. Although workable, this is not overly generous, and we would need a reconstruction algorithm that can deal with this reduced amount of projection data.

Finally, yet another possible application of CT that has come into the spotlight is intra-operative CT. Here, a patient undergoing a critical surgical procedure, such as a cerebral tumor ablation or a soft biopsy, is repeatedly imaged to verify the current status of the surgery. In contrast to intra-operative MRI, intra-operative CT does not require special non-magnetic instruments or electro-magnetic shielding. It is also much faster in the image generation

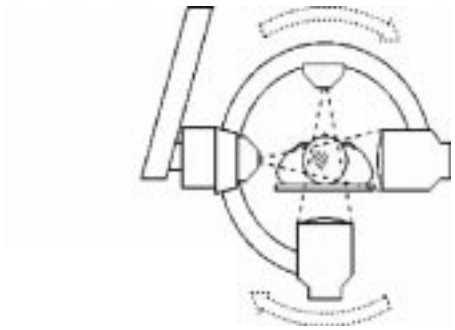


FIGURE 1.2: Biplane C-arm X-ray scanner.

process (and less expensive). However, it does object patient and personnel to radiation. Hence, it is crucial to choose both a CT modality and a reconstruction algorithm that minimize the number of acquired images and the amount of radiation dose that is needed to generate them.

This dissertation seeks to advance the field with these apparent challenges in mind. A preliminary goal can be stated as follows:

> **Devise a reconstruction framework that can faithfully and efficiently reconstruct a general volumetric object given a minimal number of projections images acquired in an efficient manner.**

In order to zero in on this preliminary goal, two main decisions had to be made:

- how are the object projections obtained, and
- what fundamental technique is used to reconstruct the object from these projections.

In both categories there are several alternatives to choose from. The next section will expand on these alternatives and justify my choice — cone-beam acquisition with algebraic reconstruction. Then, in the following section, I will present the scientific contributions of this dissertation.

1.1 Methods for CT Data Acquisition and Reconstruction

This section discusses the currently available CT-based volumetric data acquisition methods and will also expand on the two main suites of reconstruction algorithms, i.e., FBP and the algebraic techniques. The discussion concludes with the most favorable acquisition method and reconstruction technique to fulfill the preliminary goal stated above. First, I will enumerate the currently available volumetric CT imaging modalities.

1.1.1 Volumetric CT data acquisition techniques

For many years, the standard method of acquiring a volumetric CT scan has been by scan-

ning a patient one slice at a time. In this method, a linear detector array and a X-ray point source are mounted across from each other. A fan-shaped X-ray beam (such as the one shown in Figure 1.3a) traverses the patient, and the attenuated fan is stored as a 1D linear image. By spinning the source/detector pair around the patient, a series of linear images is obtained which are subsequently used for 2D slice reconstruction. A volumetric representation is obtained by advancing the table on which the patient rests after every spin, acquiring a stack of such slices. The problem with this approach is three-fold:

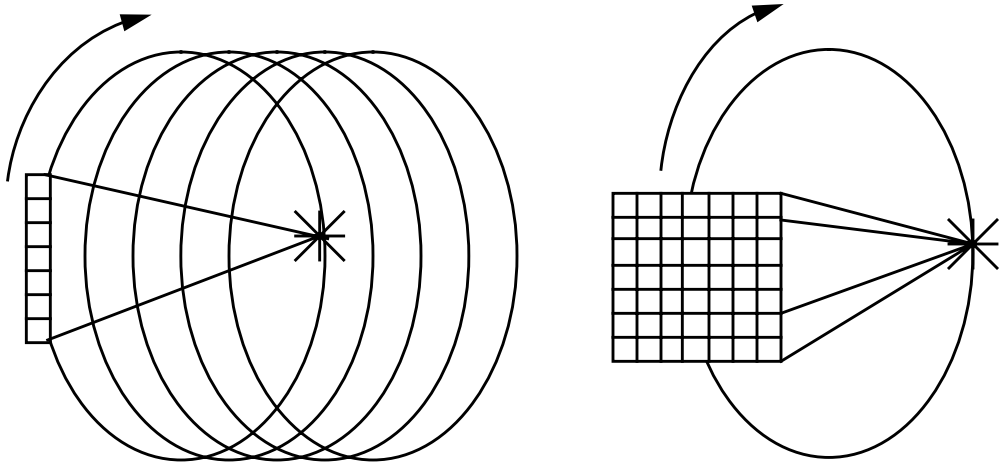
- Projection data covering patient regions between the thin slice sections is not available, which can be a problem if a tumor is located right between two slices.
- The stop-and-go table movement may cause patient displacement between subsequent slice-scans, leading to slice-misalignment in the volume.
- A volumetric scan usually takes a long time, much longer than a single breath hold, due to the setup time prior to each slice acquisition. This introduces respiratory motion artifacts into the data, another source for data inconsistency.

The first two shortcomings have been eliminated with the recent introduction of the spiral (or helical) CT scanning devices (see Figure 1.3a). Again, one works with a linear detector array, but this time the patient table is translated continuously during the scan. Special reconstruction algorithms have been designed to produce a volumetric reconstruction based on the spiral projection data [9][27]. Although spiral CT is clearly a tremendous progress for volumetric CT, some problems still remain:

- Respiratory patient motion may still cause inconsistencies in the data set. Although the volumetric scans are obtained 5 to 8 times faster than with slice-based CT, they may still take longer than a patient's breath hold.
- The emitted X-rays that fan out in a cone-shape are still under-utilized, since there is only one (or, in the more recent models, a few) linear detector array(s) that receive them. Those rays that are not used for image formation and just die out in the periphery unnecessarily contribute to the patient dose.
- 1D image data coming from the same gantry orientation, but from a different coil of the helix, are not temporally related. This makes it difficult to set up a protocol for gated imaging of moving physiological structures.

The final method leads us back to the old DSR, or better, its predecessor, the SSDSR (Single Source DSR), which adhered to the cone-beam imaging paradigm (shown in Figure 1.3). In cone-beam CT, a whole 3D dataset is acquired within only one spin around the patient. This provides for fast acquisition and better X-ray utilization, as this time a complete 2D detector array receives the cone-shaped flux of rays. The circumstance that all image data in a 2D image now originate from the same time instance enables easy gated imaging, and also provides an opportunity to use image-based methods for 3D reconstruction.

Clearly, cone-beam CT had the most prospects of meeting the set preliminary goal.



(a) (b)

FIGURE 1.3: Volumetric scanning geometries. (a) Spiral (helical) scan: the source and a 1D detector array are mounted opposite to each other and rotate in a spiral fashion many times around the patient. (b) Cone-beam scan: the source and a 2D detector array are mounted opposite to each other and rotate around the patient once.

1.1.2 Methods to reconstruct an object from its projections

In this section, I will discuss both FBP and algebraic methods in terms of their accuracy and computational cost. I will also give a brief description of both methods.

1.1.2.1 Brief description and cost of algebraic methods

A short description of the workings of algebraic methods has already been given in the beginning of this introduction. There, I mentioned that in algebraic methods an object is reconstructed by a sequence of alternating grid projections and correction backprojections. These projections and backprojections are executed for all M images in the acquired set, and possibly for many iterations I. Hence, the complexity of algebraic methods is mainly defined by the effort needed for grid projection and backprojection, multiplied by the number of iterations. Since the effort for projection and backprojection is similar, we can write the cost of algebraic methods as:

$$\begin{aligned} Cost(Algebraic) &= 2 \cdot I \cdot M \cdot Cost(Projection) \\ &= 2 \cdot a_{Alg} \cdot I \cdot M \cdot Cost(Projection) \end{aligned} \tag{1.1}$$

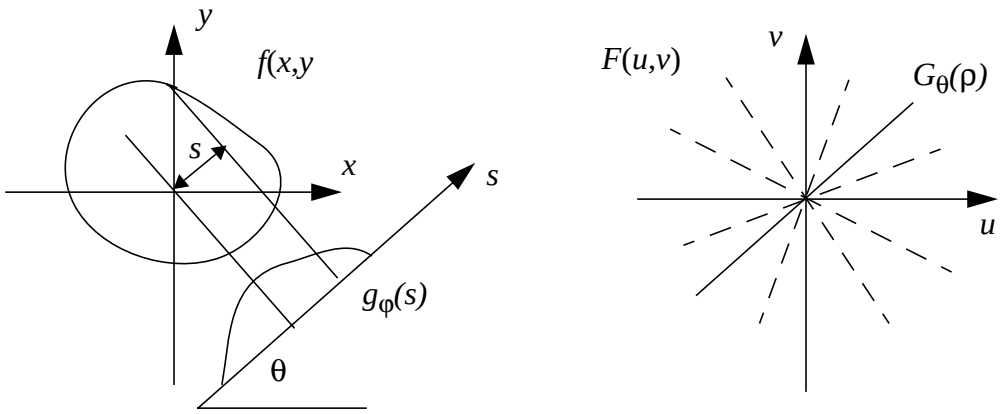
We will see the reasoning behind the second form of this equation later. For now, consider a_{Alg}=1.

1.1.2.2 Description and cost of Filtered Backprojection

The Filtered Backprojection approach is a direct method and capitalizes on the Fourier Slice Theorem [4] and the Radon transform [30] (see Figure 1.4). The Radon transform of a 2D function $f(x,y)$ is the line integral

$$g_\theta(s) = \iint f(x,y)\,\delta(x\cos\theta + y\sin\theta - s)\,dxdy \tag{1.2}$$

Thus a value at location s in a projection image $g_\theta(s)$ is given by integrating the object along a line perpendicular to the image plane oriented at angle θ. The Fourier Slice Theorem then states that the Fourier Transform (FT) of this projection image $g_\theta(s)$ is a line $G_\theta(\rho)$ in the frequency domain, oriented at angle θ and passing through the origin.



(a) Spatial domain (b) Frequency domain

FIGURE 1.4: Fourier Slice Theorem and Radon transform

Obtaining a number of projections at uniformly spaced orientations and Fourier transforming them yields a polar grid of slices in the frequency domain (shown as the dashed lines in Figure 1.4b). However, we cannot simply interpolate this polar grid into a cartesian grid and then perform an inverse Fourier transform to reconstruct the object, as is explained as follows. The 2D inverse FT (IFT) $f(x,y)$ of a signal $F(u,v)$ is given by:

$$\begin{aligned} f(x,y) &= \iint F(u,v)\exp[i2\pi(ux+vy)]\,dudv \\ &= \int_0^{2\pi} d\theta \int_0^{\infty} F(\rho,\theta)\exp[i2\pi\rho(x\cos\theta + y\sin\theta)]\,\rho\,d\rho \\ &= \int_0^{2\pi} d\theta \int_{-\infty}^{\infty} G(\rho,\theta)\exp[i2\pi\rho(x\cos\theta + y\sin\theta)]\,|\rho|\,d\rho \end{aligned} \tag{1.3}$$

The first representation is the cartesian form, the second and third is the polar form. The last from of the equation states that an object given by $f(x,y)$ can be reconstructed by scaling the polar slices $G_\theta(\rho)$ by $|\rho|$, performing a 1D IFT on them (inner integral), and finally backprojecting (="smearing") them onto the grid (outer integral). Thus we need 2 1D FT's, the multiplications with $|\rho|$, and the backprojection step. Alternatively, one can convolve the $g_\theta(R)$ by the IFT of $|\rho|$ and backproject that onto the grid. Note, that the IFT of $|\rho|$ does not exist, but approximations (using large *sinc*-type filters) usually suffice. The former method is referred to as Filtered Backprojection (FBP), the latter as Convolution Back-projection (CBP).

So far, the discussion was restricted to the 2D parallel beam case. However, as was mentioned in the previous section, modern slice-based CT scanners use the fan-beam approach, in which a point source with opposing 1D detector array are both mounted onto a rotating gantry. The (still experimental) cone-beam scanners extend the 1D linear detector array to a 2D detector matrix.

The good news is that the FBP and CBP algorithms can still be used, but with certain alterations, and, in the case of cone-beam, with certain restrictions. The diverging-beam case can be converted into the parallel-beam case by transforming the projection data as follows (see Figure 1.5).

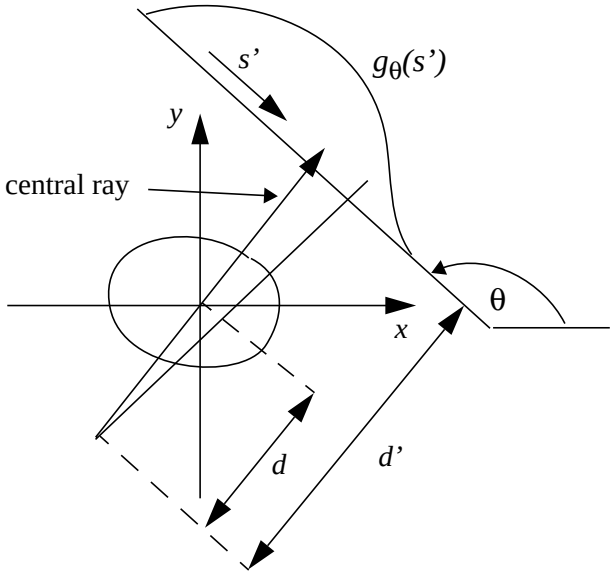


FIGURE 1.5: Fan-beam geometric arrangement

First, one must move $g_\varphi(s)$ to the origin, scaling the projection coordinates by $s=s'(d/d')$. Then, we must adjust the projection values for the path length difference due to the diverging beams by multiplying each by the term $d/\left(\sqrt{d^2+s^2}\right)$. In the case of cone-beam, we just need to consider the t coordinate as well. After the data adjustment is performed, we

filter by $|\rho|$ as usual. Due to the diverging beam geometry, the backprojection is not simply a smearing action, but must be performed as a weighted backprojection. In the following discussion, consider the 3D cone-beam case which is the most general. Even though backprojection is done in a cartesian coordinate system, our data are still in a projective coordinate system, spanned by the image coordinates *s*, *t* and the ray coordinate *r*, given by the projected length of the ray onto the central ray, normalized by *d*. The mapping of this projective coordinate system (s,t,r) to (x,y,z) is (rs,rt,r). This gives rise to the volumetric differential $dx \cdot dy \cdot dz = r^2 ds \cdot dt \cdot dr$. Thus, when backprojecting the projective data onto the cartesian grid, we must weight the contribution by $1/r^2$. The weighted backprojection integral is then written as:

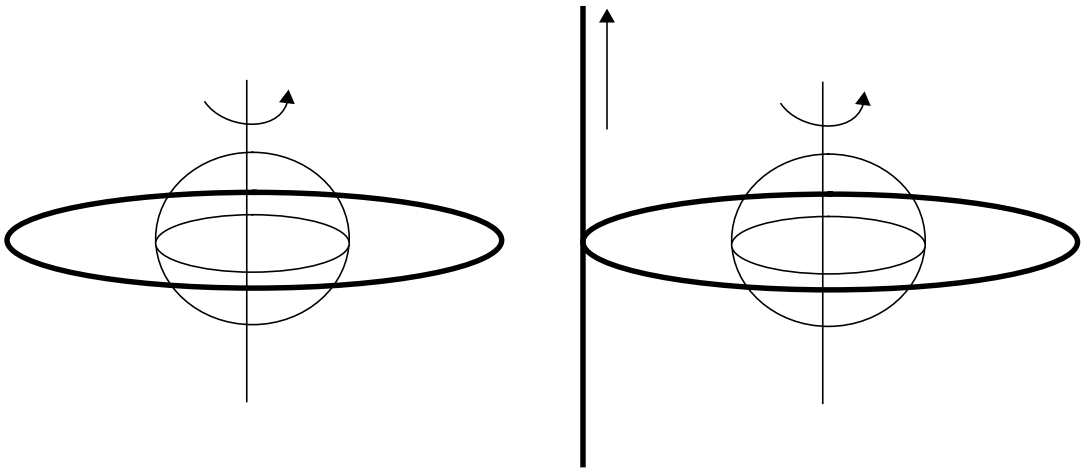
$$f(x, y, z) = \int_0^{\pi} \frac{1}{r^2} g(s, t, \theta)\, d\theta \tag{1.4}$$

In the fan-beam case, given a sufficient number of projection images (see below), an object can be reconstructed with good accuracy. The cone-beam case is, however, more difficult. Here, certain criteria with respect to the orbit of the source detector pair have to be observed. A fundamental rule for faithful cone-beam reconstruction was formulated by Tuy and can be stated as follows:

> *Tuy's Data Sufficiency Condition: For every plane that intersects the object, there must exist at least one cone-beam source point* [64].

If this condition is not warranted, certain parts of the object will be suppressed or appear faded or blurred. Rizo et.al. give a good explanation for this behavior [52]. Tuy's condition is clearly not satisfied by a single circular-orbit data acquisition (shown in Figure 1.6a). A possible remedy to this problem is to use two orthogonal circular orbits (one about the *z*-axis and one about the *x*-axis, for example) and merge the two projection sets. Note, however, that this is not always feasible. Another way of satisfying the source-orbit criterion is to scan the object in a circular orbit as usual, but supplement this scan by a vertical scan without orbital motion. This acquisition path was advocated by Zeng and Gullberg [70] and is shown in Figure 1.6b. Other trajectories have also been proposed, such as a helical scan [65] or a circular orbit with a sinusoidal variation.

Prior to a cone-beam reconstruction with the modified parallel-beam reconstruction algorithm described above, we must filter the projection images in the Fourier domain (or alternatively, convolve them in the spatial domain). If we choose to filter in the Fourier domain, we must transform and inversely transform the projection data. This is most efficiently done with a 2D FFT (Fast Fourier Transform) algorithm. The 2D FFT has a complexity of $O(n^2 \cdot \log(n))$, where n^2 is the number of pixels in a projection image. Filtering $G_\theta(\rho)$ with a ramp filter has a complexity of $O(n^2)$. Thus the resulting actual cost for the entire filtering operation (2 FFTs and the pixel-wise multiplication by $|\rho|$) is somewhat lower than the cost for a projection operation (which has a complexity of $O(n^3)$). However, a precise statement cannot be made, since we do not exactly know the constants involved in the *O*-notation. For now, we write the cost for FBP as:



(a) (b)

FIGURE 1.6: Cone-beam source orbits: (a) single circular orbit, (b) circle-and-line orbit

$$\begin{aligned} Cost(FBP) &= M \cdot (Cost(Projection) + Cost(Filtering)) \\ &= (1 + a_{FBP}) \cdot M \cdot Cost(Projection) \end{aligned} \tag{1.5}$$

Here, a_{FBP} is a factor less than 1.0. It reflects the fact that filtering is probably less expensive than grid projection.

1.1.3 Preliminary comparison of FBP and algebraic methods in terms of cost

Using equations (1.1) and (1.5), we can compare FBP and algebraic reconstruction in terms of their computational costs:

$$\frac{Cost(Algebraic)}{Cost(FBP)} = \frac{2 \cdot a_{Alg} \cdot I \cdot M_{Alg}}{(1 + a_{FBP}) \cdot M_{FBP}} \tag{1.6}$$

Clearly, if $M_{Alg}=M_{FBP}$ and $a_{Alg}=1$, algebraic methods are bound to be much slower than FBP, even when $I=1$. However, I have distinguished M_{Alg} and M_{FBP} for a reason. As was shown by Guan and Gordon in [19], theoretically $M_{Alg}<M_{FBP}$, or to be more precise, $M_{Alg}=M_{FBP}/2$. In addition, one could imagine that some of the calculations done for grid projection could be reused for the subsequent grid backprojections. This would render $a_{Alg}<1$. Taking all this into account, we may be able to afford a number of iterations $I>1$, and still be competitive with FBP. In this respect, things don't look all that bad for the algebraic methods from the standpoint of computational cost.

1.1.4 Comparison of FBP and algebraic methods in terms of quality

Let me now consider the two methods in terms of their reconstruction quality. In clinical, slice-based CT, FBP is nowadays exclusively used. This due to the computational advantages that I have outlined in the previous paragraphs. Note, however, that clinical CT scanners usually acquire more than 500 line projections per slice, which approximates the continuous form of the inverse Radon integral rather well. The true power of algebraic methods is revealed in cases where one does not have a large set of projections available, when the projections are not distributed uniformly in angle, when the projections are sparse or missing at certain orientations, or when one wants to model some of the photon scattering artifacts in the reconstruction procedure [1][30][36]. Noise as present in many clinical CT datasets is also better handled by algebraic methods [30], a fact that has been discovered by researchers in PET (Positron Emission Tomography) and SPECT (Single Photon Emission Computed Tomography) imaging as well [35][55]. Finally, algebraic methods allow *a-priory* constraints to be applied onto the shape or density of the reconstructed object, before and during the reconstruction process. In this way, the reconstruction can be influenced and guided to produce an object of better contrast and delineation than it would have without such intervention [60].

While 2D fan-beam tomographic reconstruction has been routinely used for over two decades, 3D cone-beam reconstruction is still in the research stage. As yet, there are no clinical cone-beam scanners, however, there are a number of experimental setups at academic institutions [11][47][48][60] and corporate labs (Hitachi, GE, Siemens, and Elscint). A variety of cone-beam algorithms based on FBP have also been proposed. The most notable ones are Feldkamp, Davis, and Kress [12], Grangeat [15], and Smith [59], all published in the mid and late 1980s. While the research on general 2D algebraic methods [17][19][23][25] and 2D fan-beam algebraic methods [22][36] is numerous, the published literature on 3D reconstructors using algebraic algorithms is rather sparse. One exception is the work by Matej et. al. [35], whose studies in the field of fully-3D reconstruction of noisy PET data indicate that ART, produces quantitatively better reconstruction results than the more popular FBP and MLE (Maximum Likelihood Estimation) methods. In this case, however, not a cone-beam reconstructor was used, but the projection rays were rebinned which simplified ART to the parallel-beam case. Another group of researchers has successfully applied ART for SPECT data [55] not long ago. Again, no cone-beam reconstruction was performed, rather, the data were rebinned for parallel-beam reconstruction.

However, the most popular use of 3D cone-beam ART is in 3D computed angiography. As was already touched on in a previous section of this introduction, in computed angiography one acquires images of blood-perfused structures injected with radio-opaque dye while rotating the cone-beam X-ray source-detector pair around the patient. For toxicity reasons, the dye injection time is limited and thus the number of projections that can be acquired is restricted as well. Computed angiography is a prime candidate for ART, as its projection data feature many of the insufficiencies that were noted above as being better handled by ART (as opposed to FBP): Only a limited amount of projection data is available, the projections are not necessarily taken at equi-distant angles, and the projections are usually

noisy. In addition, one can often improve the appearance and detail of the reconstructed vessels by isolating the vessels from the background after a couple of iterations, and just use these segmented volume regions in the remaining iterations [60]. For example, Ning and Rooker [47] and Saint-Felix et. al. [60] all use ART-type methods to 3D reconstruct vascular trees in the head and abdominal regions. One should keep in mind, however, that the objects reconstructed in 3D computed angiography are of rather high contrast, which poses the reconstruction problem as almost a binary one. To a lesser degree, this is also true for PET and SPECT reconstructions.

It were the promising reported qualitative advantages of algebraic methods in the limited projection data case, its better noise tolerance, and the lack of any fundamental research on algebraic methods in the up-and-coming field of general, low-contrast cone-beam reconstruction, that led me to dedicate this dissertational research to the advancement of algebraic methods in this setting. In the following, I shall give an outlook on the several scientific contributions that were achieved in the course of this endeavour.

1.2 Contributions of this Dissertation

In this dissertation, I have undertaken and accomplished the task of advancing the current state of the "ART" to a more general arena than high-contrast computed angiography. I have conceived algorithms that produce accurate 3D reconstructions for cone angles as wide as 60° and object contrasts as low as 0.5%. This just about covers the range of all cone-beam reconstruction tasks, in-vivo and in-vitro, from clinical CT to micro-tomography of biological specimen.

However, having designed a highly accurate reconstruction algorithm is only half of the story. In order to make it applicable in a clinical setting, it is also important that this algorithm produces its results in a reasonable amount of time. When this dissertation work was begun, algebraic methods were nowhere near this premise. As indicated by equation (1.6), algebraic methods have two main variables that can be attacked for this purpose:

- the number of iterations needed to converge to a solution that fits a certain convergence criterion, and
- the complexity of the projection/backprojection operations.

This dissertation provides solutions that seek to minimize both of these crucial variables, without sacrificing reconstruction quality.

Finally, with a growing number of graphics workstations being introduced in hospitals for visualization purposes, the idea is not far-fetched to use these same workstations also for 3D reconstruction of these datasets. This would lead to a better utilization of these workstations, and would certainly be a more economical solution than to build or purchase special reconstruction hardware boards. Since algebraic reconstruction mostly consists of projection/backprojection operations, a task at which these workstations are really good at, one can expect great computational speedups when porting algebraic algorithms to utilize this hardware. The final chapter of this dissertation will report on such an attempt, which

led to a speedup of over 75 compared to the software implementation, at only little decline in reconstruction quality.

Hence, the final goal of this dissertation can be defined as follows:

> **Extend the existing theory of algebraic methods into the previously unexplored domain of general, low-contrast cone-beam reconstruction. Devise techniques that provide reconstructions of high quality at low computational effort, with the ultimate aim of making ART an efficient choice for routine clinical use.**

We shall see, in the subsequent sections, how this goal was accomplished.

CHAPTER 2

ALGEBRAIC METHODS: BACKGROUND

In this chapter, the principles of algebraic methods will be described. First, I will discuss the Algebraic Reconstruction Technique (ART) [17], which is the oldest method of this kind. Then I will turn to a related method, Simultaneous ART (SART), developed later by Anderson and Kak [2] as a means to suppress noise in the reconstructions. Although this dissertation will be confined to these two variants of algebraic methods, others exist that will be briefly discussed in the final section of this chapter.

2.1 The Algebraic Reconstruction Technique (ART)

The Algebraic Reconstruction Technique (ART) was proposed by Gordon, Bender, and Herman in [17] as a method for the reconstruction of three-dimensional objects from electron-microscopic scans and X-ray photography. This seminal work was a major breakthrough, as the Fourier methods that existed in those days were highly limited in the scope of objects that they were able to reconstruct, and, in addition, were also rather wasteful in terms of their computational requirements. It is generally believed that it was ART that Hounsfield used in his first generation of CT scanners. However, as we also know, the Fourier methods matured quickly and captured ART's territory soon after.

Let me now describe the theory of ART as relevant for this dissertation. Although ART, in its original form, was proposed to reconstruct 3D objects, it represented them as a stack of separately reconstructed 2D slices, in a parallel beam geometry. For the remaining discussion, I will extend the original 2D notation into 3D, which is more convenient considering the scope of this dissertation. The extension is trivial, and can always be reduced to the 2D case.

2.1.1 ART as a system of linear equations

ART can be written as a linear algebra problem: $\boldsymbol{WV}=\boldsymbol{P}$. Here, $\boldsymbol{V}$ is the unknown ($N\times1$) column vector storing the values of all $N=n^3$ volume elements or *voxels* in the $n \times n \times n$ reconstruction grid. $\boldsymbol{P}$ is the ($R\times1$) column vector, composed of the $R=M\cdot R_m$ values of the pixels p_i in the combined set of all M projection images P_φ of R_m picture elements or *pixels* each, where the P_φ are the images obtained from the imaging device at angles φ of the X-ray detector plane (see Figure 2.1). Finally, $\boldsymbol{W}$ is the ($R\times N$) weight (or coefficient) matrix in which an element w_{ij} represents a measure of the influence that voxel v_j has on the ray r_i passing through pixel p_i. We can write $\boldsymbol{WV=P}$ in an expanded form as a system of linear equations:

$$\begin{aligned} w_{11}v_1 + w_{12}v_2 + w_{13}v_3 + \ldots + w_{1N}v_N &= p_1 \\ w_{21}v_1 + w_{22}v_2 + w_{23}v_3 + \ldots + w_{2N}v_N &= p_2 \\ &\ldots \\ w_{M1}v_1 + w_{M2}v_2 + v_{M3} + \ldots + w_{MN}v_N &= p_M \end{aligned} \tag{2.1}$$

It is clear that the weight coefficients bear a crucial role in the solution of this equation system. They are the elements that link the unknown voxel values to the known pixel values. For the solution to be accurate, each weight w_{ij} must accurately represent the influence of a voxel v_j on a ray r_i passing through pixel p_i. The first ART incarnation by Gordon, Bender, and Herman [17] represented a voxel grid by a raster of squares (see Figure 2.1). A weight w_{ij} was set to 1 if a ray r_i passed through the square of voxel v_j and was set to 0 otherwise. Later, Shepp and Logan [61] computed the actual area of intersection of the ray beam that is bounded by the rays emanated from the pixel boundaries on the image plane. It is this approach that is shown in Figure 2.1. In many current ART implementations the ray beam is reduced back to a thin linear ray r_i. However, this time a coefficient w_{ij} is determined by the length of r_i in voxel v_j. Herman et. al. have shown in their software package SNARK [24], that the calculation of the w_{ij} along r_i requires only a few additions per voxel and can be done very efficiently using an incremental Digital Differential Analyzer (DDA) algorithm (see [13]). All these implementations represent a voxel by a square (or a cubic box in 3D), which is a rather crude approximation, according to well-known principles of sampling theory [49]. The following section will expand on this issue and conclude with a voxel basis that is more appropriate than the box.

2.1.2 Choosing a good voxel basis

Consider again Figure 2.1. Here, we see that, although the originally imaged object was continuous, i.e., each point in space had a defined density value, the reconstruction only yields a discrete approximation of this object on a discrete raster. However, in contrast to the representation of Figure 2.1, this raster does not consist of equi-valued tiles, which would mean that the reconstructed object is a granulated representation of itself. Rather, it is a raster of lines in which the known values, or grid samples, are only explicitly given at

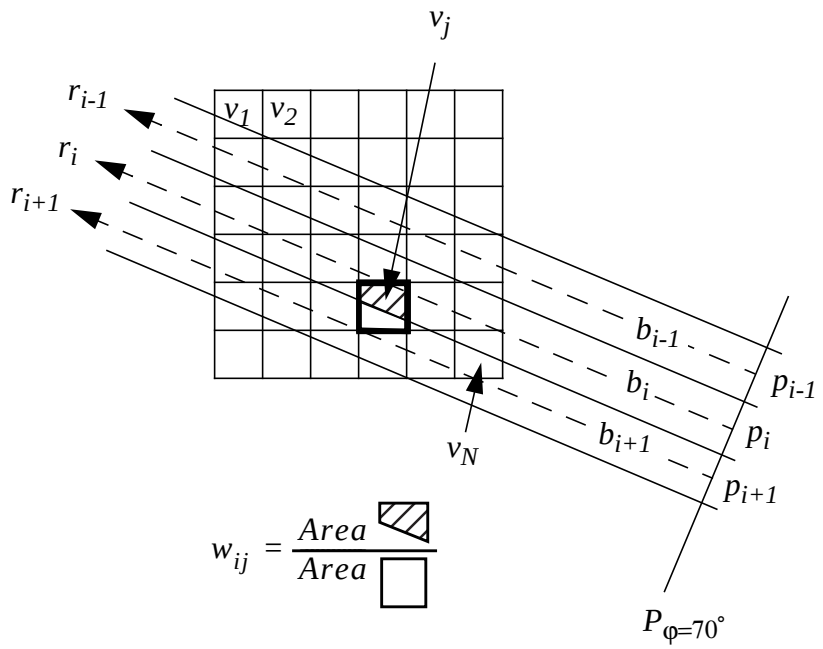


FIGURE 2.1: A beam b_i due to ray r_i passes through projection image P_φ at pixel p_i and subtends a voxel v_j. In early propositions of ART, a voxel v_j was a block (square or cube) of uniform density. A weight factor w_{ij} was computed as the subtended area normalized by the total voxel area.

the intersection of the raster lines, also called grid points. Object values at volume locations other than the grid points can be obtained by a process called *interpolation*. Hence, a continuous description of the object could (theoretically) be reconstructed by performing these interpolations everywhere in the volume. To see what is meant by interpolation, consider Figure 2.2. Here, a ray r_i is cast into the volume and samples the volume at equidistant points. Since not every point coincides with a grid point, a weighting function, represented by the interpolation kernel *h(u,v),* is centered at each sample point. The surrounding grid points that fall within the interpolation kernel's extent are integrated, properly weighted by the interpolation kernel function (here a simple bilinear function). Figure 2.2 shows how a sample value s_{ik} at a ray sample position $(X(s_{ik}),Y(s_{ik}))$ is calculated from the neighboring voxels.

The value s_{ik} is given by:

$$s_{ik} = \sum_j h\left(X(s_{ik}) - X(v_j), Y(s_{ik}) - Y(v_j)\right) \cdot v_j \tag{2.2}$$

The entire ray sum for pixel p_i is then given by the sum of all s_{ik} along the ray r_i:

$$p_i = \sum_k \sum_j h\left(X(s_{ik}) - X(v_j), Y(s_{ik}) - Y(v_j)\right) \cdot v_j \tag{2.3}$$

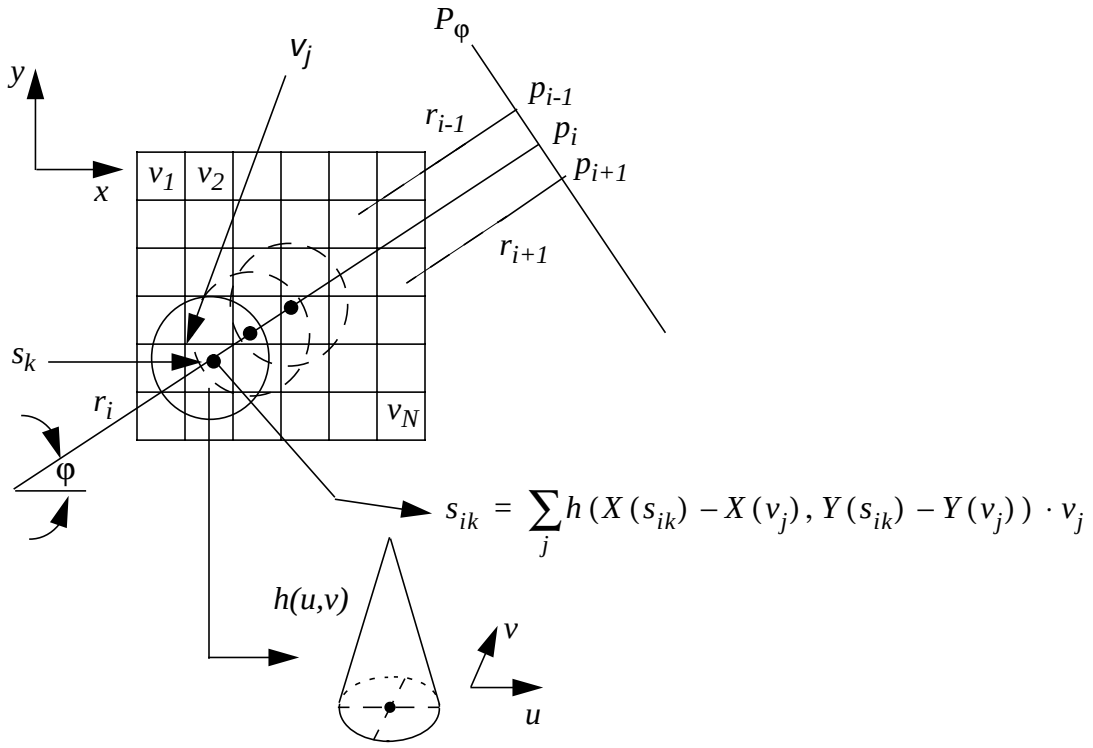


FIGURE 2.2: Interpolation of a ray sample value s_{ik} located at $(X(s_{ik}),Y(s_{ik}))$ in the reconstruction grid. All those discrete reconstruction grid points v_j that fall into the extent of interpolation kernel $h(u,v)$ centered at $(X(s_{ik}),Y(s_{ik}))$ are properly weighted by the interpolation kernel function at $(X(s_{ik})-X(v_j),Y(s_{ik})-Y(v_j))$ and summed to form the value of sample s_{ik} of ray r_i.

This ray sum is a discrete approximation of the ray integral:

$$p_i = \int\left(\sum_j h(X(s_i) - X(v_j), Y(s_i) - Y(v_j)) \cdot v_j\right) ds_i \tag{2.4}$$

We can reorder this equation into:

$$p_i = \sum_j v_j \cdot \int h(X(s_i) - X(v_j), Y(s_i) - Y(v_j))\, ds_i \tag{2.5}$$

This is shown in Figure 2.3. We see the similarity to the equations in (2.1). Here, a pixel was given by:

$$p_i = \sum_k v_j \cdot w_{ij} \tag{2.6}$$

Hence, the weights are given by the integrals of the interpolation kernel along the ray:

$$w_{ij} = \int h(X(s_i) - X(v_j), Y(s_i) - Y(v_j))\, ds_i \tag{2.7}$$

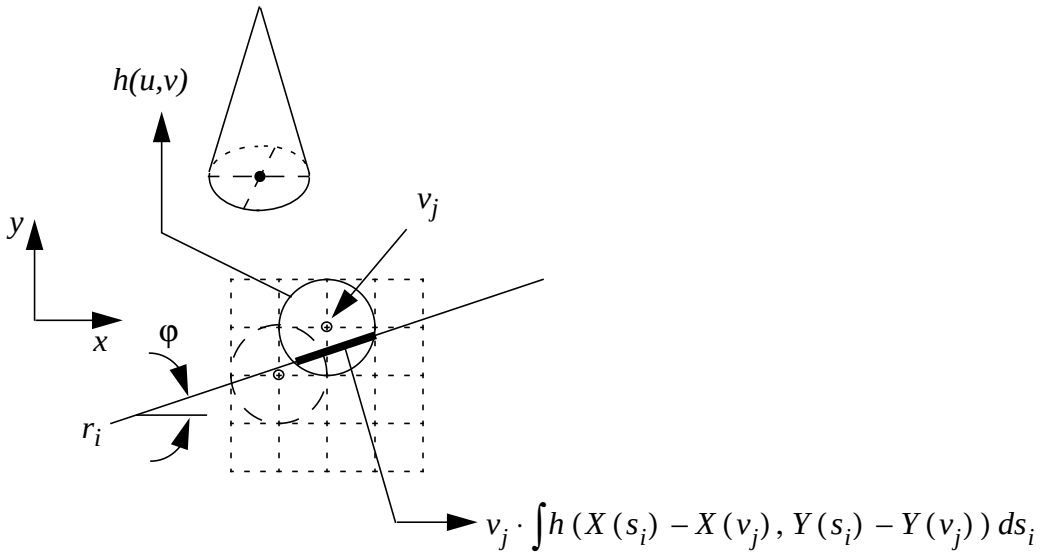


FIGURE 2.3: A ray r_i at orientation φ enters the kernel of voxel v_j and integrates it along its path.

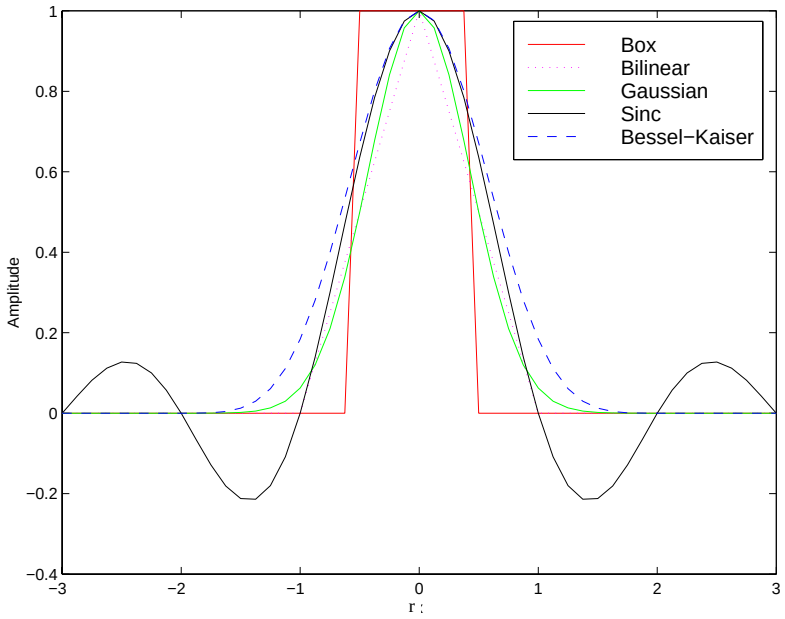
Obviously, the better the interpolation kernel, the better the interpolation, and the more accurate the weight factor. Figure 2.4 shows some popular interpolation kernels, both in the spatial (Figure 2.4a) and in the frequency domain (Figure 2.4b). (We will return to these plots in a few paragraphs.) The frequency plot tells us how good a filter is. Generally, when reconstructing a function from discrete samples, we want the filter to have a high amplitude for frequencies < $0.5/T_g$ and a low amplitude for frequencies > $0.5/T_g$, where $1/T_g$ is the grid sampling rate. We want that because discrete functions have their spectra replicated as *aliases* at multiples of $1/T_g$, and reconstructing with an interpolation filter means multiplying the discrete signal's frequency spectrum with the spectrum of the filter.

To clarify these statements, let me analyze the interpolation process a little more in detail. The interpolation of a sample value at any location in a discrete grid can be decomposed as follows:

- Reconstruction of the continuous function *f* from the discrete grid function f_s by convolving *f* with the interpolation filter *h*.
- Resampling of the reconstructed continuous function *f* at the sample's location $(X(s_{ik}),Y(s_{ik}))$.

Consider now Figure 2.5a. Here, we see the frequency spectrum of the discrete grid signal, F_s. The spectrum has a main lobe centered at f=0, and aliases, replicated at a frequency of $1/T_g$. When reconstructing *f* from f_s, we want to recover just the main lobe and none of the aliases. However, real-life interpolation filters will always include some of the aliases into

(a)



(b)

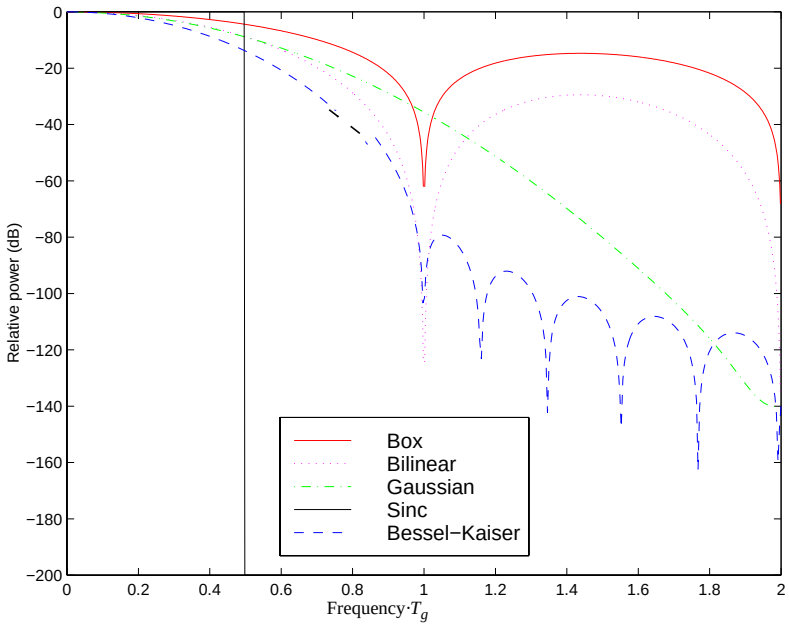


FIGURE 2.4: Some interpolation filters: (a) spatial plots, (b) frequency plots.

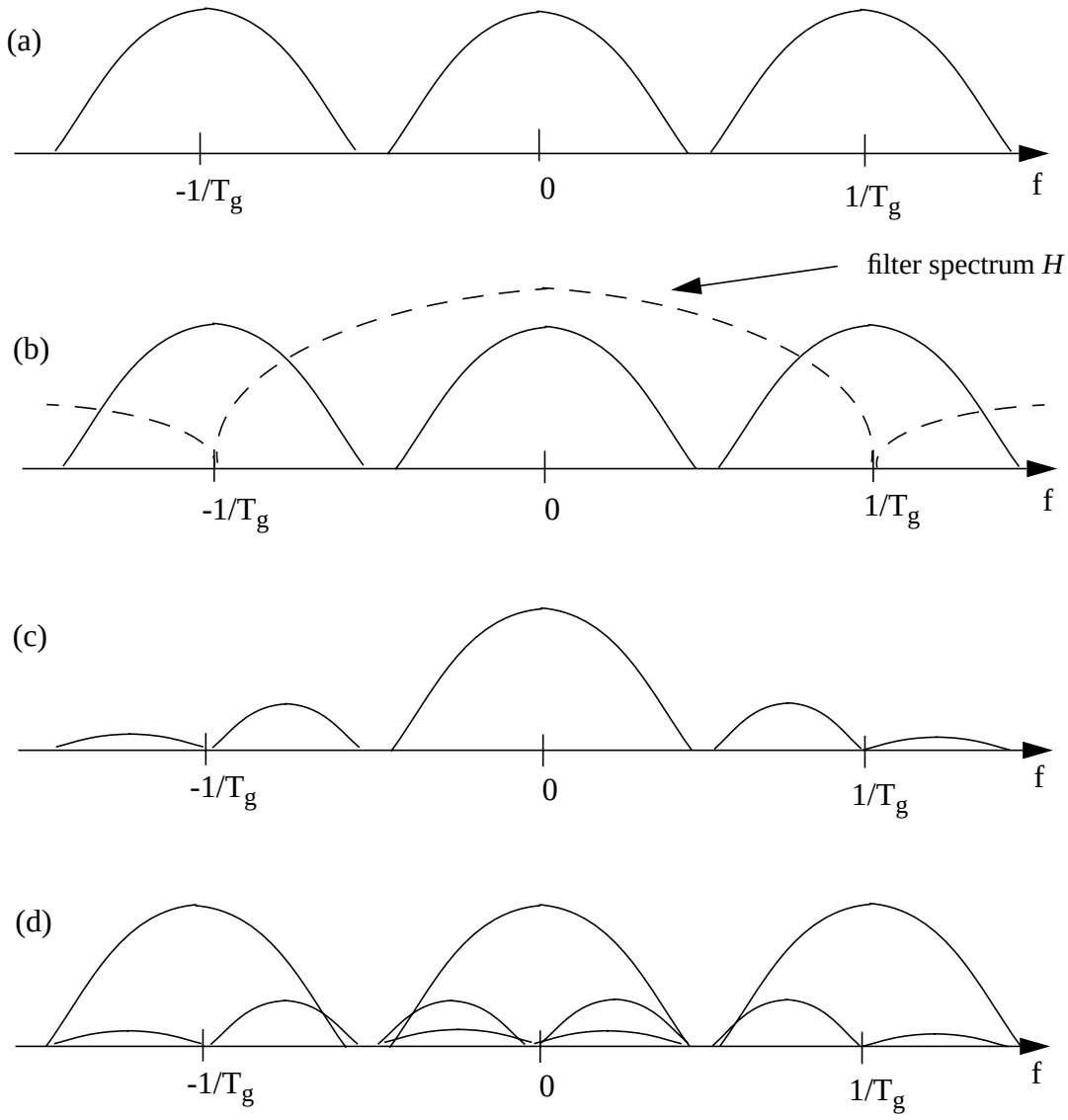


FIGURE 2.5: Interpolation of a discrete signal (solid line), frequency domain: (a) Spectrum of the original discrete signal, F_s, replicas (*aliases*) of the signal spectrum are located at multiples of $1/T_g$. (b) Multiplication with the spectrum of the (non-ideal) interpolation filter H. (c) Resulting frequency spectrum F of the interpolated signal. Note that some of the signal's aliases survived due to the imperfect interpolation filter. (d) Re-sampled interpolated signal. The aliases in the signal replicas in the sidelobes show up in the main lobe. The composite signal is irrecoverably distorted.

the reconstructed function. Let me now illustrate what the decomposition of the interpolation process in the spatial domain, as shown above, translates to in the frequency domain:

- Reconstruction of f from f_s is equivalent to the multiplication of the discrete signals's spectrum F_s with the spectrum of the interpolation filter, H. This process is shown in Figure 2.5b. Note the remaining (attenuated) signal aliases in F, shown in Figure 2.5c. These aliases have disturbing effects (such as ghosts and false edges) in f. (This form of aliasing is called *post-aliasing*).
- Resampling of f is equivalent to replicating the reconstructed spectrum F at a frequency $1/T_g$. Note, in Figure 2.5d, that the attenuated signal aliases that survived the reconstruction process now reach into neighboring lobes in the new discrete signal. Thus, the re-sampled discrete signal has irrecoverable artifacts (called *pre-aliasing*).

A far more detailed description of this process is given in [69]. Now, as we know more about the effect of a good interpolation filter, let me return to Figure 2.4. Here, we see a few interpolation filters, both in the spatial domain and in the frequency domain. The ideal interpolation filter is the *sinc* filter, as its box-shape in the frequency domain blocks off all signal aliases. The sinc filter passes all frequencies $< 0.5 \cdot 1/T_g$ (the *passband*) unattenuated, and it totally blocks off all frequencies $> 0.5 \cdot 1/T_g$ (the *stopband*). The sinc filter, however, has an infinite extent in the spatial domain, and is therefore impractical to use. On the other extreme is the box filter. It has an extent of 0.5 in the spatial domain, and has the worst performance in blocking off signal aliases in the stopband. It is that boxfilter that is used by the ART implementations mentioned before. We now easily recognize that this filter, although very efficient due to its small spatial extent, is probably not the best choice for reconstruction. Figure 2.4 also shows the frequency spectra and spatial waveforms of the bilinear filter, a Gaussian filter $2\text{^}(-4 \cdot r^2)$, and a Bessel-Kaiser filter. The last two filters have a spatial extent of 4.0 and are also radially symmetric in higher dimensions. This will prove to be an advantage later.

The family of Bessel-Kaiser functions were investigated for higher dimensions by Lewitt in [32] and used (by the same author) for ART in [31] and [33]. The Bessel-Kaiser functions have several nice characteristics:

- They are tunable to assume many spatial shapes and frequency spectra. Consider again the frequency plots in Figure 2.4b. We observe that the spectrum of the Bessel-Kaiser function has a main-lobe that falls off to zero at the grid sampling frequency $\omega=1/T_g$, and returns with many fast decaying side-lobes. The tuning parameters control the placement of the zero-minima between the mainlobe and the sidelobes. They also control the rate of decay. Since the signal's first sidelobe is at maximum at a frequency $\omega=1/T_g$, we would want to place the zero-minimum of the filter's spectrum at that location for best anti-aliasing effects. The circumstance that the filter decays fast in the stopband suppresses all of the signal's higher-order sidelobes to insignificance.
- Another characteristic of the Bessel-Kaiser function is that they have, in contrast to the Gaussian function, a finite spatial extent (here a radius of 2.0). Truncation effects, that possibly appear with a necessarily truncated Gaussian, are thus avoided.

- Finally, the Bessel-Kaiser function has a closed-form solution for the ray integration.

Due to Lewitt's convincing arguments, and also due to a very efficient implementation (described later), I have solely used Bessel-Kaiser functions as the voxel basis in the dissertational research presented here. However, it should be noted that other voxel bases have also been used in the past: Andersen and Kak represented a voxel by a bilinear kernel [2], and I (in [44]) chose a Gaussian kernel in the early days of this research.

For convenience and completeness, I shall repeat some of the results derived in [32] for the Bessel-Kaiser function in higher dimensions. First, the generalized Bessel-Kaiser function is defined in the spatial domain as follows:

$$\begin{aligned} b^m(r) &= \frac{\left(\sqrt{1-(r/a)^2}\right)^m I_m\left(\alpha\sqrt{1-(r/a)^2}\right)}{I_m(\alpha)} && 0 \le r \le a \\ b^m(r) &= 0 && otherwise \end{aligned} \tag{2.8}$$

Here, a determines the extent of the function (chosen to be 2.0), and α is the taper parameter that determines the trade-off between the width of the main lobe and the amplitude of the side lobes in the Fourier transform of the function. (When α=0, we get the box function of width 2a.) The function $I_m()$ is the modified Bessel function of the first kind and order m. We set m=2 to get a continuous derivative at the kernel border and everywhere else in the kernel's extent.

The corresponding Fourier transform is given as:

$$\begin{aligned} B_n^m(\omega) &= \frac{(2\pi)^{n/2} a^n \alpha^m I_{n/2+m}\left(\sqrt{\alpha^2-(2\pi a\omega)^2}\right)}{I_m(\alpha)\left(\sqrt{\alpha^2-(2\pi a\omega)^2}\right)^{n/2+m}} && 2\pi a\omega \le \alpha \\ B_n^m(\omega) &= \frac{(2\pi)^{n/2} a^n \alpha^m J_{n/2+m}\left(\sqrt{(2\pi a\omega)^2-\alpha^2}\right)}{I_m(\alpha)\left(\sqrt{(2\pi a\omega)^2-\alpha^2}\right)^{n/2+m}} && 2\pi a\omega \ge \alpha \end{aligned} \tag{2.9}$$

Here, n is the dimension of the function to be reconstructed. If we reconstruct 2D slices, n=2, for volumetric 3D reconstruction n=3. The function $J_{n/2+m}()$ is the Bessel function of the first kind and order $n/2+m$. We see that, depending on n, different values of α are necessary to place the zero-minimum between the filter's mainlobe and its first sidelobe at $\omega=1/T_g$. We can calculate that for a 2D kernel α=10.80, and for a 3D kernel α=10.40.

Finally, the Abel transform $p^m(r)$ (i.e., the X-ray projection) of the generalized Bessel-Kaiser function $b^{(m,\,\alpha)}(r)$ can be shown to be proportional to $b^{(m+1/2,\,\alpha)}(r)$:

$$p^m(r) = \frac{a\sqrt{2\pi/\alpha}}{I_m(\alpha)}\left(\left(\sqrt{1-(r/a)^2}\right)^{m+1/2} I_{m+1/2}\left(\alpha\sqrt{1-(r/a)^2}\right)\right) \tag{2.10}$$

2.1.3 The Kaczmarz method for solving the system of linear equations

Let me now return to equation system (2.1) and see how it can be solved. The first problem is that we cannot assume that $R=N$, actually in most applications $R \neq N$. In some cases we can enforce $R=N$ by adding interpolated subpixels in the projection images or by adding projections. But nevertheless, the enormous size of $\boldsymbol{W}$ prohibits the use of matrix inversion methods to solve $\boldsymbol{WV}=\boldsymbol{P}$ for $\boldsymbol{V}$. When $R>N$, least squares methods could be applied, but this also proves to be computationally impractical if N is large. In the most common case, where $N>R$, many solutions exist that satisfy $\boldsymbol{WV}=\boldsymbol{P}$. It is the goal of ART to find that solution that represents the closest approximation to the object function from which the projection images were obtained.

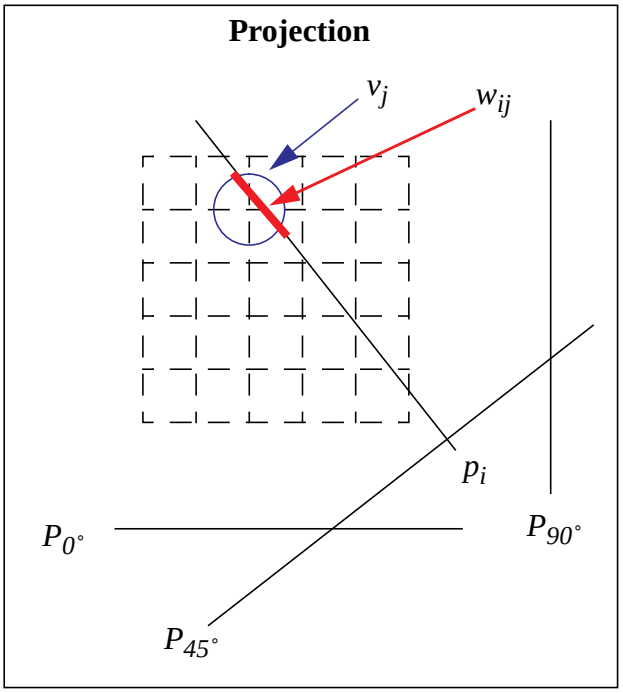
Setting aside the issue that in most cases $R \neq N$, it turns out that usually noise and sampling errors in the ART implementation do not provide for a consistent equation system anyhow. Thus, Gordon, Bender, and Herman [17] selected an iterative scheme proposed by Kaczmarz [29] as early as 1937 to solve the equation system. In this procedure, one starts from an initial guess for the volume vector, $\boldsymbol{V}=\boldsymbol{V}^{(0)}$, and selects at each iteration step k, $k>0$, one of the equations in (2.1), say the one for p_i. A value $p_i^{(k)}$ is measured which is the value of pixel i computed using the voxel values as provided by the present state of the vector $\boldsymbol{V}=\boldsymbol{V}^{(k)}$. A factor related to the difference of $p_i^{(k)}$ and p_i is then distributed back onto $\boldsymbol{V}^{(k)}$ which generates $\boldsymbol{V}^{(k+1)}$ such that if a $p_i^{(k+1)}$ were computed from $\boldsymbol{V}^{(k+1)}$, it would be closer to p_i than $p_i^{(k)}$. Thus, we can divide each grid update into three phases: a projection step, a correction factor computation, and a backprojection step.

The correction process for one element of $\boldsymbol{V}$, i.e. v_j, can be expressed by:

$$v_j^{(k+1)} = v_j^{(k)} + \lambda \frac{p_i - \sum_{n=1}^{N} w_{in} v_n^k}{\sum_{n=1}^{N} w_{in}^2} w_{ij} \tag{2.11}$$

where λ is the *relaxation factor* typically chosen within the interval (0.0,1.0], but usually much less than 1.0 to dampen correction overshoot. This procedure is performed for all equations in (2.1). After all equations have been processed, in some order, and the grid has not converged to a solution fitting some convergence criterion, the just described procedure is repeated for another iteration, and so on. See also Figure 2.6 for a detailed illustration of the ART algorithm.

Figure 2.7 shows a geometric illustration of the Kaczmarz algorithm, solving a system of two linear equations. The axes represent the two unknowns v_1 and v_2, while the lines represent the two equations (the slopes are given by the weight factors). The solution is the intersection of the two lines. One starts with an initial guess $\boldsymbol{V}^{(0)}$. It can be shown that a $\boldsymbol{V}$ correction is equivalent to dropping a line perpendicular to the line representing the selected equation, starting at the current state of $\boldsymbol{V}^{k}$, and intersecting this line with the line



Equation

$$v_j^{(k+1)} = v_j^{(k)} + \lambda \frac{p_i - \sum_{n=1}^{N} w_{in} v_n^k}{\sum_{n=1}^{N} w_{in}^2} w_{ij}$$

Algorithm

Initialize volume
Until convergence
  Select a pixel p_i from one of the scanner projections $P\varphi$
  Projection: Compute line integral through p_i
  Correction factor: Subtract line integral from the value of p_i
  Backprojection: Distribute correction onto grid

FIGURE 2.6: The ART algorithm. The colors in the equation refer to the corresponding highlighting colors in the algorithm.

representing the selected equation. This intersection point represents the new state of the vector $\boldsymbol{V}$, $\boldsymbol{V^{k+1}}$. We then select a new equation and perform this procedure again. It is obvious that this procedure will converge faster the more orthogonal the lines (i.e., the equations) are. While orthogonalization using the Gram-Schmidt procedure is computationally not feasible, attempts exist to pairwise orthogonalize the equation system [51]. These attempts, however, have not become popular. A way to achieve an *approximate* pairwise orthogonalization is to carefully plan the order in which the equations are applied. It is desirable that subsequently applied equations are as orthogonal as possible. In this manner, one can reach the solution point faster. Chapter 3 of this dissertation will elaborate more on this issue. For more than two unknowns, the equations become hyperplanes. Note, however, that once there are more then two equations, the lines (hyperplanes) may not intersect at one single point. This could be due to noise in the projections (i.e., equations) or inaccurate estimation of the weight factors. In this case, many (approximate) solutions exist, and the relaxation factor λ, the initial guess, and the equation ordering potentially all have a great effect on which solution of the many approximate solutions ART will come closest to.

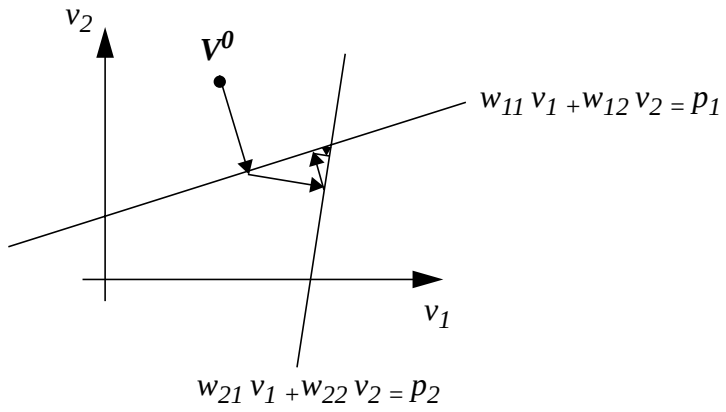


FIGURE 2.7: Kaczmarz' iterative procedure for solving systems of linear equations. This example solves 2 equations for 2 unknowns, v_1 and v_2. Each equation is represented by a line. The solution is the intersection of the two lines. To find the solution point, one starts with an initial guess, selects one of the equation lines, and drops a line perpendicular to it, starting at the initial guess. The intersection point then becomes the new guess. The procedure is continued until the solution point is reached, or one gets sufficiently close to it. In higher dimensions, the lines become hyperplanes.

As was hinted on in the previous paragraph, the initial guess $\boldsymbol{V^{(0)}}$, the order in which the equations are chosen in corrective process, and the accuracy of the weight factors w_{ij} all have a significant impact on what solution $\boldsymbol{V^{(s)}}$ the vector $\boldsymbol{V}$ converges to. It was proven by Tanabe [63] that the Kaczmarz approach will generally converge to a vector $\boldsymbol{V^{(s)}}$ such that $|\boldsymbol{V^{(0)}} - \boldsymbol{V^{(s)}}|$ is minimized. However, I have found that, in practice, the initial guess is not so

important. It is more the accuracy of the weights, the magnitude of the relaxation factor λ, and the order in which equations are selected, that play a large role both in speed of convergence and in the quality of the reconstruction that is converged to. I will expand on all these topics in the subsequent chapters of this dissertation.

2.1.4 Efficient grid projection

The sum terms in the nominator and denominator of equation (2.11) require us to compute the integration of a ray across the volume. However, we do not need to compute the ray integrals via the slow process of raycasting. As we have already seen in Section 2.1.2, a voxel weight factor corresponds to the integration of the ray across the interpolation kernel function. So all we really need to do is to accumulate the individual kernel contributions for a given ray (squared, in the denominator of equation (2.11), or scaled by the voxel value, in the nominator). Very efficient methods exist to build a ray integral from these individual voxel contributions. If the interpolation kernel is radially symmetric, we can pre-integrate the kernel integrals $\int h(s)\,ds$ (compare equation (2.7)), often analytically, into a lookup-table. This lookup table is called the kernel *footprint*. In case of the Bessel-Kaiser function we may use equation (2.10) to calculate the kernel footprint lookup-table. We then map all voxel footprints to the screen, squared or scaled, where they accumulate into a projection image. It was Lee Westover who first proposed this approach in [66]-[68] and termed it *splatting*. He, however, did not use it for algebraic methods, but for volume rendering [28], where it is very popular up to this day. Matej and Lewitt considered splatting later for block-iterative, parallel-beam ART [33].

Alternatively, one can also use rays to intersect the lookup tables in volume space, again scale or square the indexed value, and accumulate the density integrals ray by ray. This approach was proposed much earlier than the splatting algorithm by Hanson and Wecksung in [22], this time for algebraic reconstruction. Note, however, that none of these approaches were ever extended to 3D cone-beam reconstruction. They were just used for 2D fan-beam reconstruction (in [22]) or for parallel-beam geometries (in [33]). Chapter 5 of this dissertation will expand both the splatting and the ray-driven approach to 3D cone-beam reconstruction and will also give very efficient implementations.

Both of these approaches capitalize on the fact that by placing an interpolation kernel h at each grid voxel j and scaling it by the grid voxel's value v_j, we obtain a field of overlapping interpolation kernels that reconstructs the discrete grid function f_s into a continuous function f. Thus, both a projection of these voxel basis functions onto the screen and a traversal of the grid by interpolating rays yield (potentially) the same (X-ray) image. However, the former approach has the advantage that the footprint functions can be pre-integrated analytically or by good quadrature methods, which yields a highly accurate, almost analytical ray integral summation on the screen. Raycasting, on the other hand, interpolates the volume at discrete locations, which necessarily gives only an approximation to the analytical ray integral. How good this approximation will be depends on the ray step size and the quadrature method. Hence, the splatting approach, be it voxel-driven or ray-driven, is the best projection method, both for accuracy and for efficiency reasons.

So far, I have only discussed the projection step. But, as it turns out, backprojection is performed in a similar way, only that here the voxels receive (corrective) energy, scaled by their weight factors, instead of emitting it.

2.1.5 Why ART needs fewer projections than FBP

This question was answered by Guan and Gordon in [19] for the 2D parallel beam case. Let me repeat it here.

Usually one reconstructs on a square voxel grid with a sidelength of n voxels, thus the number of grid voxels $N=n^2$. Also, we generally assume a circular reconstruction region. Voxels outside this region may be ignored. In this case we have $(1/4)\,\pi n^2$ unknown voxel values and n pixels per 1D image. For equation system (2.1) to be determined, the number of projection images M_{ART} has to be:

$$M_{ART} = \frac{(1/4)\,\pi n^2}{n} = \frac{\pi n}{4} = 0.785 \cdot n \tag{2.12}$$

Let's see how many projections are needed for Filtered Backprojection (FBP). The sampling interval in Fourier space is at least $\Delta\omega=1/nT_g$, and the maximum frequency is given by $\omega_{max}=1/(2T_g)$. Due to the polar sampling in frequency space, the density of samples decreases as we go outward in the polar grid. To ensure a sampling rate of at least $\Delta\omega$ everywhere in the polar grid, even at the boundary, the angular spacing between the projections (i.e., the Fourier slices) in frequency space needs to be:

$$\Delta\varphi_P = \frac{\Delta\omega}{\omega_{max}} = \frac{2T_g}{nT_g} = \frac{2}{n} \tag{2.13}$$

In order to provide adequate sampling in the periphery, one must oversample in the interior frequency regions. The number of projections M_{FBP} is then:

$$M_{FBP} = \frac{\pi}{\Delta\varphi_P} = \frac{\pi n}{2} = 1.57 \cdot n \tag{2.14}$$

Thus, $M_{Alg}=M_{FBP}\,/\,2$.

In 3D reconstruction, one usually reconstructs on a cubic voxel grid, again with a sidelength of n voxels. Thus the number of grid voxels $N=n^3$. Also, for a 3D single-orbit reconstruction we generally assume a spherical reconstruction region. In this case we have $(1/6)\,\pi n^3$ unknown voxel values and $(1/4)\,\pi n^2$ relevant pixels per image. For the equation system (2.1) to be determined, the number of projection images M_{ART3D} has to be:

$$M_{ART3D} = \frac{(1/6)\,\pi n^3}{(1/4)\,\pi n^2} = 0.67n \tag{2.15}$$

This means that for n=128, a total of 86 projection images is required.

The widely accepted rule of thumb is that FBP requires at least n projection images [54] for

good image quality. The previous calculation showed that ART requires about half of that. As a matter of fact, this is true for all algebraic methods, and not just ART.

We have now determined some of the factors in the cost equation (1.6), repeated here for convenience:

$$\frac{Cost\,(Algebraic)}{Cost\,(FBP)} = \frac{2 \cdot a_{Alg} \cdot I \cdot M_{Alg}}{(1 + a_{FBP}) \cdot M_{FBP}}$$

With the new knowledge that we just gained, we can rewrite this equation:

$$\frac{Cost\,(Algebraic)}{Cost\,(FBP)} = \frac{2 \cdot a_{Alg} \cdot I}{(1 + a_{FBP}) \cdot 2} = \frac{a_{Alg} \cdot I}{(1 + a_{FBP})} \tag{2.16}$$

We will find out about the other factors later.

2.2 Simultaneous ART (SART)

In 1984, Andersen and Kak noticed (in [2]) that a voxel may be traversed by many rays during one iteration. They showed that ART's method of updating the grid for every ray caused "striping" in the reconstruction. These correction stripes in turn were blamed for the high overall level of noise that existed in the ART reconstructions back in those days. The authors argued further, that if, in contrast to ART, the correction terms were accumulated for each voxel and the grid was only updated after each iteration, as was done in the Simultaneous Iterative Reconstruction Technique (SIRT) [14], then this "striping" was apparently suppressed. The problem with SIRT, however, was the long time it needed for convergence. In an attempt to combine the positive aspects of both techniques, i.e. ART and SIRT, Andersen and Kak developed the Simultaneous ART (SART) [2]. In this method, grid correction is not performed for each single pixel separately. Instead, a whole projection image is computed first (say at the orientation of $P\varphi$), and then each voxel in the reconstruction grid is corrected by an accumulated correction term that is due to all pixels in $P\varphi$. Since ambiguities manifested as noise are introduced into the reconstruction image if the correction terms are simply added, a weighted sum of the contributions is used to update a voxel.

Mathematically, this can be expressed as follows:

$$v_j^{(k)} = v_j^{(k-1)} + \lambda \frac{\sum_{p_i \in P_\varphi} \left(\frac{p_i - \sum_{n=1}^{N} w_{in} v_n^{(k-1)}}{\sum_{n=1}^{N} w_{in}} \right) w_{ij}}{\sum_{p_i \in P_\varphi} w_{ij}} \tag{2.17}$$

Note that equation (2.17) differs from equation (2.11) in two ways: First, the correction terms imposed on a particular voxel v_j by a set of neighboring projection image pixels p_i are pooled using the respective w_{ij} factors to weight their influence. Second, we do not use the square of the w_{ij} in the sum of weights in the denominator of the correction factor term. Although ART in the spirit of Kaczmarz' method requires the sum of *squared* weights, SART seems to be more consistent with the concept of viewing ART as an inverse volume rendering technique [28], where a volume is iteratively reconstructed by a sequence of summed volume renderings (the projections) and inverse summed volume renderings (the backprojections). In this context, it makes sense to normalize each ray integral by the physical length of the ray, represented by $\sum w_{in}$. The SART procedure is illustrated in Figure 2.6 in greater detail.

Later research [35], and also my own experience [38], has indicated, that it was probably not the image-based SART correction procedure itself that has prevented the noise-like artifacts from occurring, rather, it was Andersen and Kak's use of a better interpolation kernel, i.e. the bilinear function, as a voxel basis. Still, SART has many advantages, due to its image-based approach:

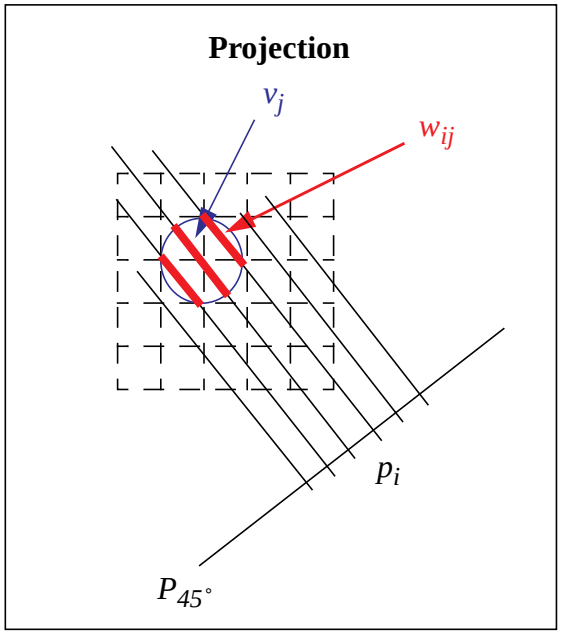
- It proves to be the better method for cone-beam reconstruction (unless ART is modified, as we will see later).
- It lends itself very well for a graphics hardware-accelerated approach.

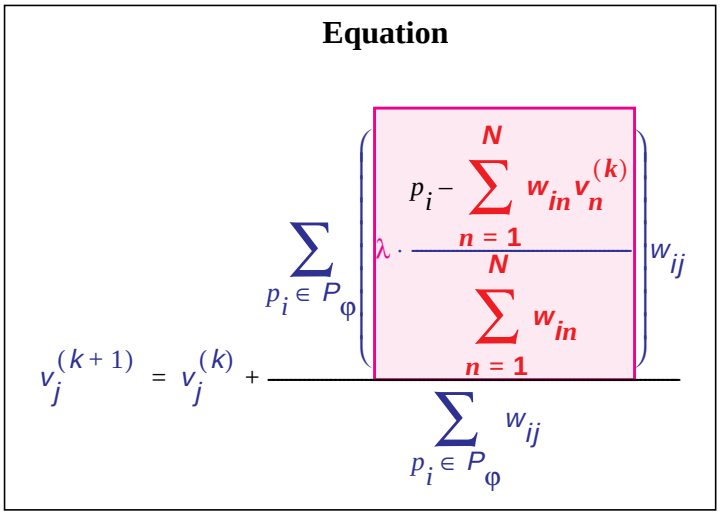
Both of these advantages have been discovered in the course of this dissertation research. However, it was also discovered that SART is slightly slower than ART in software, due to the voxel-based pooling of correctional updates.

2.3 Other Algebraic Reconstruction Methods

Finally, I would like to mention other existing algebraic methods. I have already touched on the Simultaneous Iterative Reconstruction Technique, proposed shortly after ART by Gilbert [14]. In this approach, the voxels are only updated after the corrections due to all R projection pixels have been computed. These corrections are then properly weighted and normalized (similar to SART) before adding them to the voxels. SIRT's approach is rather slow and has not gained a wide popularity due to that fact. The circumstance that the SIRT-cousin SART does not perform quality-wise any better for cone-beam reconstruction than (a modified version of) ART (as demonstrated in this dissertation) may serve as evidence that SIRT probably will not do so either.

Algebraic reconstruction methods that update the grid separately for each ray (such as traditional ART) are called row-action methods, since each computation step is equivalent to a dot product of one row in the weight matrix $\boldsymbol{W}$ and the object vector $\boldsymbol{V}$. In contrast, methods that correct the grid based on the projection of a block of rays involving a multi-element set of such dot products are called block-iterative methods. These methods are discussed for example in [7]. The block-iterative methods combine two or more (almost) parallel hyperplanes (see Figure 2.7) to reach a convergence faster. SART and SIRT are not strictly





Algorithm

Initialize volume
Until convergence
  Select a projection $P\varphi$
  Image projection: Compute line integrals through all p_i of $P\varphi$
  Correction image: For all p_i, subtract line integrals from p_i
  Image backprojection: Distribute corrections onto grid

FIGURE 2.8: The SART algorithm. The colors in the equation refer to the corresponding highlighting colors in the algorithm.

members of the traditional block-iterative algebraic methods. However, they also combine equations (due to one projection or due to all projections) in an attempt to update the volume with a correction that is a is based on many equations, and not just one.

Along with (additive) ART, Gordon, Bender, and Herman also proposed Multiplicative ART, or MART [17]. The correction equation is:

$$v_j^{(k+1)} = \lambda \frac{p_i}{\sum_{n=1}^{N} w_{in} v_n^k} v_j^k \tag{2.18}$$

This method has the advantage that, in contrast to ART, a voxel can never be corrected to values less than zero (which is not in the solution space). However, MART has never reached the popularity of ART, and the original authors found that while ART seems to minimize the overall variance of the reconstruction, MART maximizes the entropy.

CHAPTER 3

A GLOBALLY OPTIMIZING PROJECTION ACCESS ALGORITHM

As was mentioned in the previous chapter, the order in which the projections are applied in the iterative reconstruction procedure has a crucial effect both on reconstruction quality and speed of convergence. This chapter will present a new projection ordering method, termed the Weighted Distance Scheme (WDS), that is superior to the existing methods of this kind. I will start with some background on this topic, describe the existing methods and their shortcomings, and then turn to the specifics of the new method. Note, that for the remainder of this chapter, an iteration constitutes a sequence of grid corrections in which all available projections are utilized exactly once.

3.1 Introduction

Due to the problems associated with the linear equation system (2.1), many solutions may exist. It is thus the goal of ART to converge to the solution that represents the closest approximation to the object function from which the projection images were obtained. In this respect, it has been known for quite some time [23][51] that both the quality of the approximation and the rate of convergence of the iterative reconstruction procedure depends, among other factors, on the order in which the projections are selected for grid correction.

A number of researchers have pointed out [18][25] that it is desirable to order the projections in such a way that subsequently applied projections are largely uncorrelated. This means that consecutively applied projections must have significantly different angular orientations. Indeed, it is intuitive to recognize that if subsequently selected projections are chosen at similar orientations, one tends to overly bias the reconstruction with respect to that viewing angle without adding much new information to the grid. Clearly, doing so prolongs the time for convergence and may also drive the approximate solution away from the desired solution.

While all previously proposed ordering schemes take great care to space far apart consecutively chosen projections, they somewhat neglect the problem of optimizing the selection in a global sense. It is the argument of this chapter that in the process of selecting a newly applied projection all, or at least an extended history of, previously applied projection orientations must be taken into account and weighted by their time of application.

In the following section, I will give a brief overview of previous work on projection access ordering methods. Then, in Section 3.3, I will present a novel projection ordering method, the Weighted Distance Scheme (WDS), which heuristically optimizes the angular distance of a newly selected projection with respect to the complete sequence of all previously applied projections (including those applied in the previous iteration) or any continuous, time-wise adjacent subset thereof. Finally, Section 3.4 gives a numerical comparison of the new method with respect to previous methods.

3.2 Previous Work

In order to minimize correlation in projection access it seems advantageous to arrange the projections such that (Postulate 1):

a. a series of subsequently applied projections is evenly distributed across a wide angular range,

b. at no time is there an angular range that is covered more densely than others.

All of the existing methods tend to be strong in one of the two aspects, but weaker in the other. However, none of the previous methods comments on how one should proceed with the projection selection at iteration boundaries. It is clearly necessary to also include projections applied in previous iterations into the selection process. A smooth transition between iterations is warranted if the selection scheme is continuous across iteration boundaries.

Denoting M as the total number of projections in the set and $0 \le j \le M-1$ to be the number of projections already applied, Hamaker and Solmon [21] demonstrate in a fundamental treatment that a "good" permutation τ of the ordering of the M projections is obtained when the minimum angle among $(\tau(l)-\tau(k)) \cdot 180^\circ/M$, $0 \le l \le j$, $0 \le k \le j$, $l \ne k$ is maximized for each j. Postulate 1 could be regarded as a more heuristic interpretation of this finding.

Many implementations have used a fixed angle for projection spacing: SART, for example, uses a constant angle of 73.8° when M=100. For most M there is, however, no fixed angle that satisfies the criterion of maximizing the minimum angle for all $1 \le j \le M-1$.

An alternative method that uses variable angles between subsequent projections was proposed by Herman and Meyer [25]. It is based on the prime number decomposition (PND) of M. This method, however, requires that M be non-prime. The same authors also refer to work performed by van Dijke [10] who concluded that, among all schemes he tried, a random projection permutation gave the best results. However, we may prefer an ordering scheme that is more controllable and deterministic than a random number generator.

More recently, Guan and Gordon [18] presented, what they termed, the Multilevel Access Scheme (MLS). This method works best when the number of projections is a power of 2, but can also be used, with minor modifications, in the general case. The following description is for the simple case of M being a power of 2: First, for level one and two, the method chooses the projections at 0˚, 90˚, 45˚, and 135˚. All subsequent levels L=3, .., $log_2 M$ contain 2^L views. The projection order at level L is computed by simply going through the list of all applied projections at levels $l<L$ and adding $M/2^L$ to their projection index. This method clearly covers all angular regions evenly over time, but may not always maximize the angle between subsequent projections.

Figure 3.1a shows, for M=30, the obtained permutation of the first 15 selected projection views when the scheme of PND is applied. Figure 3.1b shows the permutation for the MLS method under the same conditions. We observe that PND tends to cluster the projections around certain viewing directions. This may not be advantageous in light of our earlier comments with regards to an even spread of the applied projections around the reconstruction cycle. As expected, the MLS method generates a permutation that conforms more closely to this criterion.

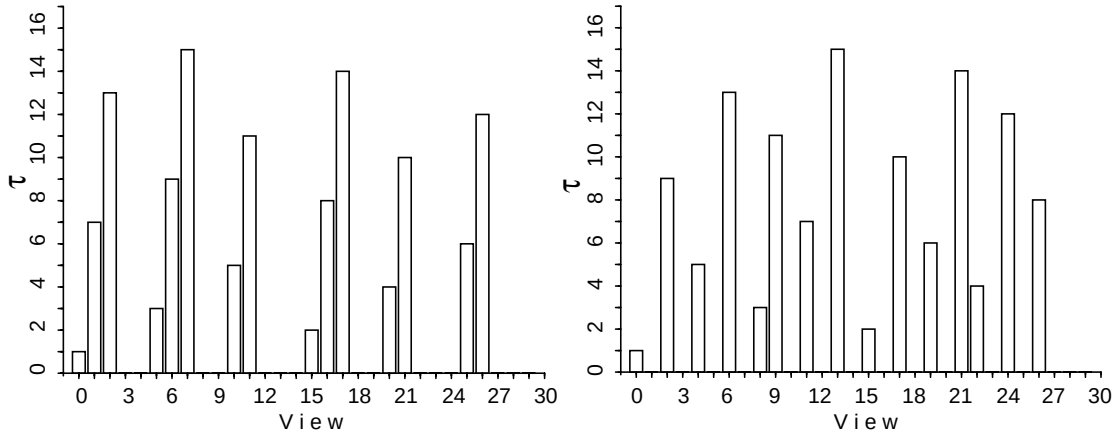


FIGURE 3.1: Projection order permutation τ for M=30 projections. For illustrative purposes only the first 15 ordered views are shown. The height of a bar corresponds to a view's place in the ordered sequence τ (For instance, in (a) projection 5 is applied as the third projection after view 0 and view 15). The graphs shown are for: (a) Prime number decomposition (PND), (b) Multilevel access scheme (MLS).

3.3 The Weighted Distance Projection Ordering Scheme

Let me now introduce a permutation scheme that seeks to enforce both parts of Postulate 1. It is designed to maintain a large angular distance among the whole set of used projections

while preventing clustering of projections around a set of main view orientations. The method selects, from the pool of unused projections, that projection that optimizes both the angular spacing and the spread with respect to the complete set or a recent subset of all previously applied projectional views. Hereby it takes into account that more recent applied projections should have a stronger influence in the selection process than projections that have been applied earlier in the reconstruction procedure.

The algorithm implements the permutation of the projection ordering as a circular queue Θ. The length S of Θ depends on how much of the projection access history is to influence the selection process. Since due to the decaying weighting function the influence of early projections diminishes to insignificance over time, early projections can be replaced by later ones in Θ as time evolves. In my implementation, I chose $S=M$: A projection's influential power fades to zero after M more projections have been selected. Note, however, that the number of influential projections represented by S can be chosen arbitrary large.

The projections P_i, $0 \le i \le M-1$, are assumed to be equally spaced by an angle $\varphi = 180^\circ / M$ in the interval $0 \le i \cdot \varphi < 180^\circ$. The first projection inserted into Θ (at position $\Theta[0]$) and applied in the reconstruction procedure is always the projection at orientation angle $\varphi = 0^\circ$, i.e. P_0. A list Λ is maintained that holds the L projections not yet used in the current iteration. At the beginning of each iteration, Λ is filled with all M projections. When a projection P_i is selected from Λ by our selection criterion, it is removed from Λ and inserted into Θ. Insertion can occur within two phases: the initial filling phase of Θ and the update phase of Θ. Let Q denote the number of projections currently in Θ. In the filling phase, $Q<S$ and subsequently used projections are added to consecutive positions in Θ. In the update phase, $Q=S$ and Θ now functions as a circular queue: The oldest projection in Θ is overwritten by the newly selected projection and ceases to influence the selection process. See Figure 3.2 for an illustration of the algorithm in pseudo-code.

```
InitCircularQueue(Θ);   /* circular queue Θ is initially empty */

while not converged
    FillProjectionPool(Λ);   /* all projections are available (again) */

    for i = 1 ... M
        P=SelectProjection(Λ,Θ);    /* select a projection P from Λ based on the distance to
                                       all previous applied projections in queue Θ */
        RemoveFromList(P,Λ);  /* P is removed from Λ and no longer available for this
                                 iteration */

        AddToEndOfCircularQueue(P,Θ);    /*P goes at the end of Θ, the oldest item in
                                           Θ falls out if Θ is full (Q = S) */

        ApplyProjection(P);    /* perform grid projection and backprojection for P */
```

FIGURE 3.2: Pseudo-code to illustrate insertion/removal of projections into/from circular queue Θ and list Λ within the basic framework of ART.

Let me now describe the objective function used to select the next projection from Λ (i.e., the routine SelectProjection() in Figure 3.2): First, for each P_l in Λ, $0 \le l < L\text{-}1$, the weighted mean μ_l of the "repulsive forces" exerted onto P_l by the projections P_q in Θ, $0 \le q \le Q\text{-}1$, $\Theta[q] \neq \Lambda[l]$, is computed. The *repulsive force* is considered a measure of how close one projection is to another, it decays linearly with increasing distance between two projections. A smaller repulsive force results from a larger spacing of two projections. The minimal distance of two projections P_l and P_q is given by:

$$d_{lq} = Min\,(|l-q|, S-|l-q|)\,. \tag{3.1}$$

The weighted mean of the repulsive forces acting on a projection P_l, denoted by μ_l, is given by:

$$\mu_l = \sum_{q=0}^{Q-1} w_q \cdot (S/2 - d_{lq}) \,/\, \sum_{q=0}^{Q-1} w_q \tag{3.2}$$

where a weight factor $w_q = (q+1)/Q$. The weighting ensures that projections applied more recently have a stronger repulsive effect than projections applied earlier in the reconstruction procedure.

However, using the distance criterion alone does not achieve the goals of Postulate 1. It was observed that a newly selected projection could minimize the distance criterion by being very distant to some projections in Λ, but at the expense of being very close to others. This circumstance lead to a situation where projections were selected from one of several clusters in a cyclic fashion, a condition we strived to avoid. In order to eliminate the large distance fluctuations that gave rise to this behavior I added a second measure to be minimized: the weighted standard deviation of the distances d_{lq}, σ_l :

$$\sigma_l = \sqrt{\sum_{q=0}^{Q-1} w_q \cdot \left(d_{lq} - \overline{d}_l\right)^2 \,/\, \sum_{q=0}^{Q-1} w_q} \tag{3.3}$$

Here $\overline{d}_l = \sum d_{lq}/Q$ is the average distance of P_l to the P_q. Maintaining a small σ_l of the projection distances prevents projections from clustering into groups of angular viewing ranges.

We then normalize the μ_l to a range of [0,1]:

$$\tilde{\mu}_l = \frac{\mu_l - \underset{0 \le k < L\text{-}1}{Min}(\mu_k)}{\underset{0 \le k < L\text{-}1}{Max}(\mu_k) - \underset{0 \le k < L\text{-}1}{Min}(\mu_k)} \tag{3.4}$$

The normalized standard deviations $\tilde{\sigma}_l$ are computed from the σ_l in a similar fashion.

Finally, we select that projection $P_l \in \Lambda$ to be applied next that minimizes the weighted L_2-norm:

$$D_l = \tilde{\mu}_l^2 + 0.5 \cdot \tilde{\sigma}_l^2 . \tag{3.5}$$

Experiments indicated that a factor of 0.5 to weigh σ_l^2 seemed to yield the best results for a wide range of *M*.

3.4 Results

Table 3.1 gives the projection access orders for all six ordering schemes discussed in the previous sections (*M*=30): Sequential Access (SAS), Fixed Angle at 66.0° (FAS), Prime Number Decomposition (PND) [25], Random Access (RAS) [10], Multilevel (MLS) [18], and Weighted Distance (WDS).

In order to compare all presented methods with regards to Postulate 1, we define a discrete 2D space that is spanned by the projection index number and the projection access time instance. This space, called the Projection Access Space (PAS), is "sampled" by the projection ordering methods in a N-rooks fashion, i.e., no line and column can be sampled twice since the tuple (projection index, access time) is one-to-one and onto. Figure 3.3 shows the sampling patterns for the six projection ordering schemes for *M*=30.

By visual inspection, FAS, PND, MLS and WDS all seem to have a fairly uniform sample distribution. However, to capture the quality of the distributions in light of Postulate 1 in a more quantitative way, a descriptive metric is needed. Part a) of this postulate calls for a uniform distribution of the sample points in PAS. We may measure this property by sliding a square box across the PAS, counting the number of sample points inside the box at each box position, and computing the standard deviation of all counts. If all areas are equally sampled then the standard deviation should be small. We performed this analysis for *M*=30, 80, and 100, respectively. The sliding square box was dimensioned to capture about 10% of the samples on the average. Thus the box sizes were 10×10 for *M*=30, 25×25 for *M*=80, and 30×30 for *M*=100. Part b) of the postulate was designed to prevent the clustering of projections around a few angular ranges. We evaluate this property after half of the projections (i.e. *M*/2) have been applied. For this purpose, we slide a *M*/2×4 sized box along the vertical direction of the PAS, aligned to the left PAS border. Ideally, all boxes should have an equal number of applied projections in them (i.e., 2). Again, we count the number of incidences within each box and compute the standard deviation of all counts. A larger standard deviation is evidence for an uneven distribution caused by clustering.

The results of this analysis are listed in Table 3.2. We see, while PND performs well with respect to projection access uniformity, it tends to cluster projections into angular groups. On the other hand, MLS tends less to clustering (except for *M*=80), but exhibits inferior projection access uniformity. Table 3.2 also shows that WDS behaves equally well in both categories, access uniformity and cluster-freeness, where it is better or at least as good as any other method investigated.

Finally, we tested all projection access schemes on the low-contrast 2D Shepp-Logan phan-

SAS	FAS	PND	RAS	MLS	WDS
0	0	0	3	0	0
1	11	15	24	15	15
2	22	5	7	8	25
3	3	20	2	22	7
4	14	10	1	4	19
5	25	25	15	19	1
6	6	1	0	11	12
7	17	16	10	26	23
8	28	6	27	2	5
9	9	21	29	17	17
10	20	11	6	9	28
11	1	26	13	24	10
12	12	2	19	6	21
13	23	17	20	21	3
14	4	7	26	13	14
15	15	22	22	28	26
16	26	12	4	1	8
17	7	27	25	16	18
18	18	3	5	7	29
19	29	18	8	23	6
20	10	8	28	5	24
21	21	23	9	20	13
22	2	13	12	12	2
23	13	28	21	27	20
24	24	4	16	3	11
25	5	19	14	18	22
26	16	9	17	10	4
27	27	24	23	25	16
28	8	14	18	14	27
29	19	29	11	29	9

TABLE 3.1 Projection access orders for all six ordering schemes (M=30).

tom as described in [61] and shown in Figure 3.5. 80 projections of 128 rays each were computed analytically from the mathematical description of the ellipses that make up the phantom. In the reconstruction procedure, λ was set to a fixed value of 0.30 and an interpolation kernel based on the previously mentioned Bessel-Kaiser function was used.

For estimation of the reconstruction error we use the normalized *root mean squared error measure* [23]:

$$error = \left[\sum_{i=1}^{N} (o_i - v_i)^2 \Big/ \sum_{i=1}^{N} (o_i - \bar{o})^2 \right]^{\frac{1}{2}} \tag{3.6}$$

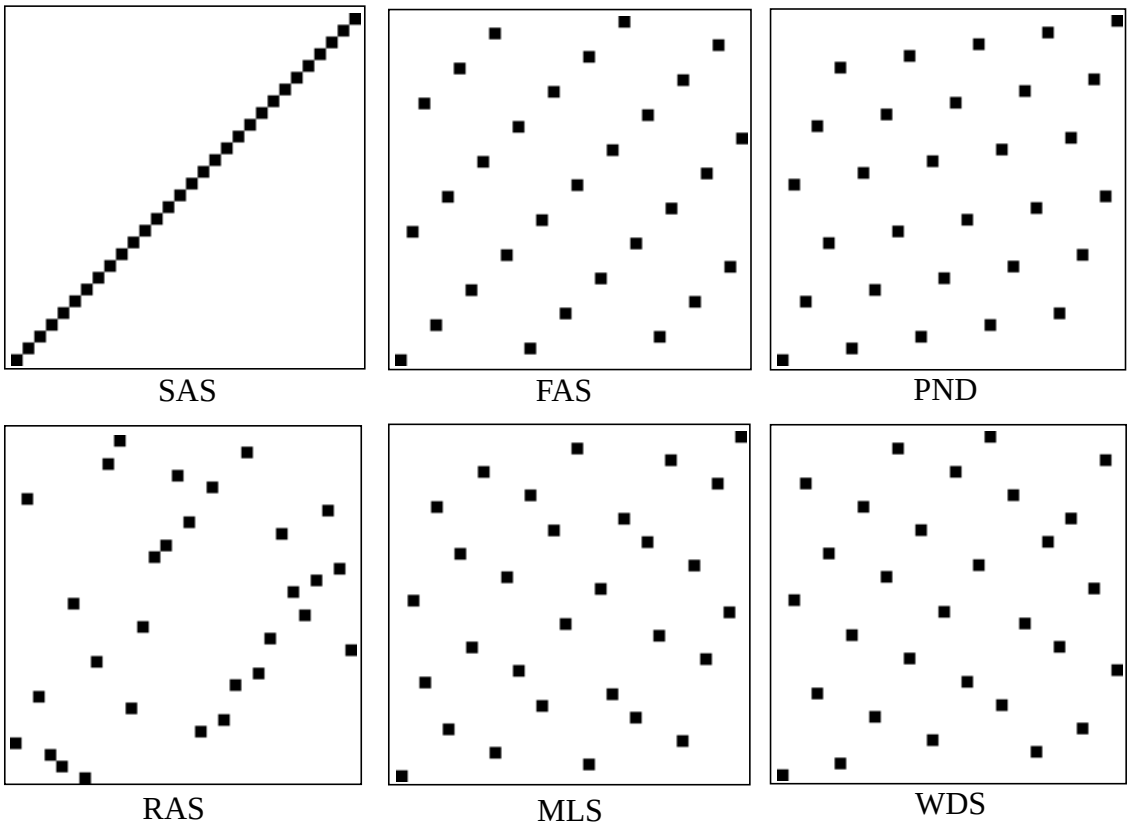


FIGURE 3.3: Sampling patterns in projection access space for the six projection ordering schemes (M=30). In all plots, the time coordinate runs from left to right, while the projection index runs from bottom to top.

Here, o_i is the value of pixel v_i in the original Shepp-Logan phantom. This error is plotted in Figure 3.4 for all six permutation schemes (M=80) for a) the entire head and b) the region around the three small tumors in the bottom half of the phantom (see Figure 3.5). We would also like to compare the various methods in terms of the level of the noise-like reconstruction artifacts. For this purpose, we compute the variance within the union of two circular regions to the right and left, respectively, of the two brain ventricles (i.e., the dark elliptical structures in the head center, see Figure 3.5). These circular regions are homogeneous in the original phantom. The variance is plotted in Figure 3.4c for all ordering methods.

The behavior of the selection process at iteration boundaries was as follows: While SAS, FAS, PND, and MLS apply their previous access order anew (which is probably the strategy used by the respective authors), RAS clears its list of used projections and chooses the next projection at random, thus generating a different access order for each iteration. WDS by design generates a different access sequence in every iteration as well, however, with the additional constraint of optimal fit with regards to the previous sequence. At this point it should also be mentioned that some authors (such as [1]) linearly increase the relaxation factor λ from a small value at initial projections to a larger fixed value for use in later grid

Projection access scheme	Access uniformity			Access clustering		
	M=30	M=80	M=100	M=30	M=80	M=100
SAS	3.251	8.224	9.859	0.333	0.216	0.195
FAS	0.650	0.808	0.905	0.133	0.075	0.066
PND	0.600	0.694	0.733	0.115	0.071	0.063
RAS	1.316	2.124	1.983	0.156	0.107	0.103
MLS	0.721	0.720	0.758	0.094	0.087	0.063
WDS	0.600	0.704	0.700	0.094	0.064	0.058

TABLE 3.2 Standard deviations of box counts for three projection set magnitudes (M=30, 80, and 100) to measure projection access uniformity and clustering. (The fixed angle used in FAS for M=30, 80, and 100 was 66.0˚, 69.75˚, and 73.8˚, respectively.)

corrections. To eliminate the effects of yet another variable in the comparison process, we chose to use a fixed value of λ throughout the reconstruction procedure. Here, λ=0.3 was found to have the best convergence properties for all access ordering schemes.

For the full head section, all five non-sequential ordering schemes reach their minimum error at about the same time, (i.e., at the end of the fourth iteration). However, this error is smallest with WDS. Hence, even though WDS does not provide a faster convergence to its minimum-error solution, the solution is more accurate compared to the solutions obtained with the competing methods at all iterations (at least until the overall error increases again). Although the difference in error is rather small for the full head section, it is considerably more significant for the isolated tumor area. This could be viewed as evidence that by using WDS small object detail can be better recovered. It is also interesting to note that all ordering schemes reach their error minimum for the tumor area about one iteration later than for the full head section. We further observe that images produced by WDS have the least amount of noise-like reconstruction artifacts, with RAS being the closest competitor.

Figure 3.6 shows the reconstruction results obtained after three iterations using the various projection ordering schemes. Even though the numerical results are to a great extent visible in the images, we would like to note that, for the human vision system, larger numerical error does not always translate to a more visible artifact. For instance, larger random noise is much more gracefully tolerated by the human eye than a small but periodic noise pattern such as ringing. From Figure 3.6 we observe that, apart from SAS, which after 3 iterations is far from reaching its minimal error, FAS and PND have considerably more reconstruction artifacts than MLS, RAS, and WDS. The artifacts are least noticeable with RAS and WDS, while the contrast for the small tumors is best with MLS and WDS.

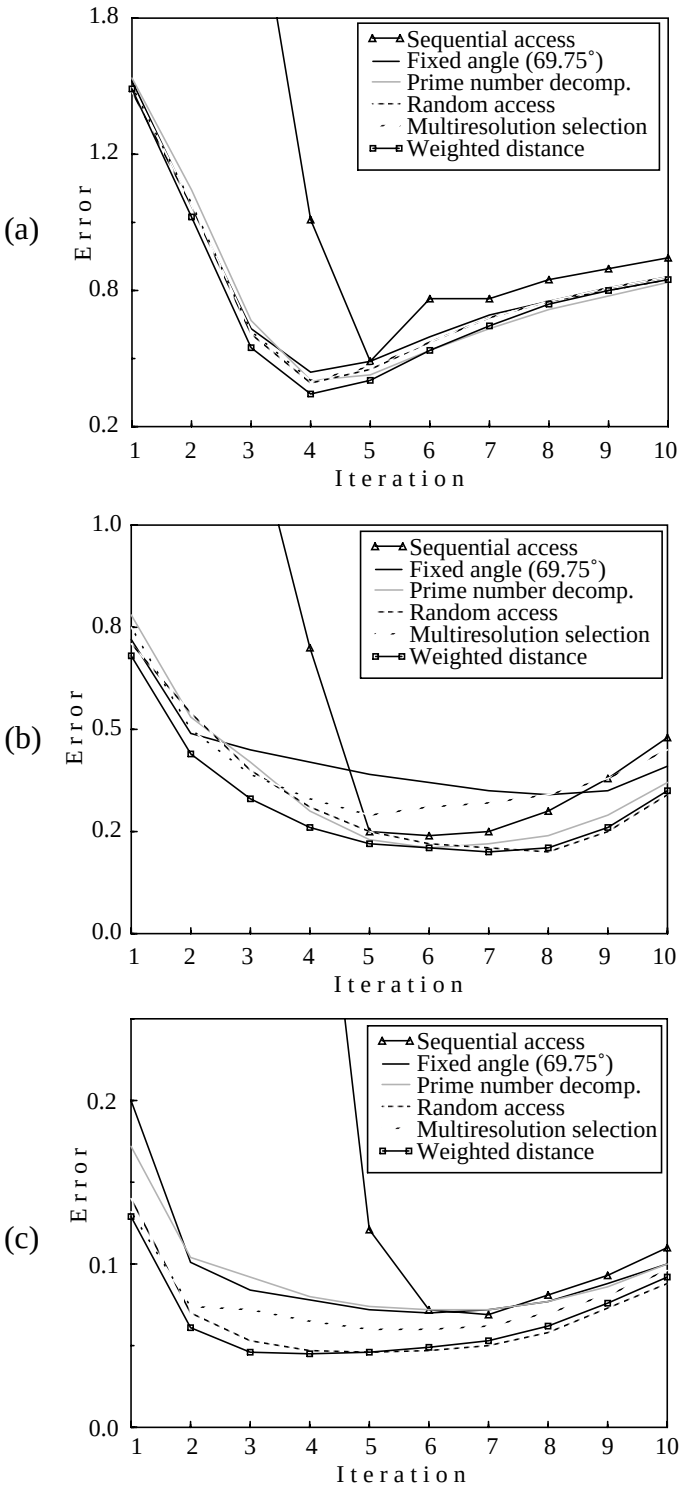


FIGURE 3.4: Reconstruction errors for Shepp-Logan phantom (80 projections of 128 pixels each, 128×128 grid): (a) Entire head segment, (b) Area with the three small tumors only, (c) Reconstruction noise.

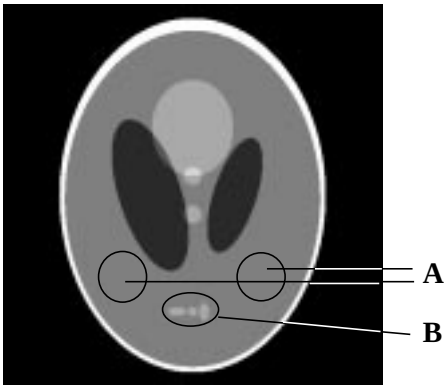


FIGURE 3.5: Original Shepp-Logan phantom with circled areas indicating the regions in which the error measurements were performed: Regions *A* were used to evaluate the level of noise artifacts, *B* is the region in which the reconstruction error for the three small tumors was measured. Notice that the pixel intensities in this and all other images involving the Shepp-Logan brain phantom presented in this dissertation were thresholded to the interval [1.0,1.04] to improve the displayed contrast of the brain features. For example, the contrast between the three small tumors in region B and the background matter is only 0.5% of the full density range.

In conclusion, WDS exhibits more uniform projection access space sampling than existing methods and delivers more accurate reconstructions. In particular, fine detail is more faithfully recovered and the degree of noise-like reconstruction artifacts is smaller. Both of these features are important as they reduce the chance of ambiguities for both the clinician and computerized image analysis systems.

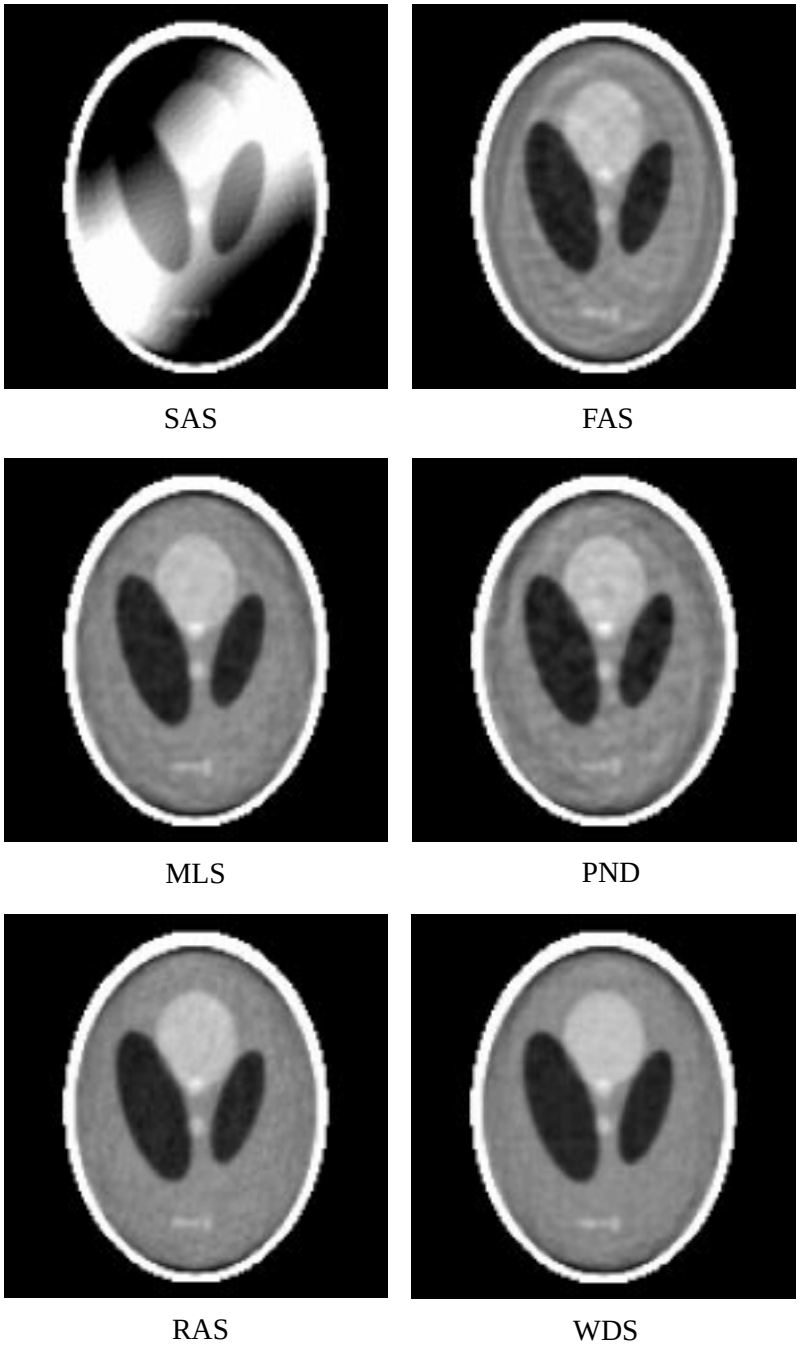


FIGURE 3.6: Reconstruction of the Shepp-Logan phantom after 3 iterations on a 128×128 grid using 80 projections of 128 rays each and λ=0.3 for the six projection access schemes.

CHAPTER 4

CONE-BEAM RECONSTRUCTION WITH ART: ACCURACY ISSUES

In this chapter, we will analyze algebraic methods for the low-contrast cone-beam setting, a scenario that has not been studied much in the past. In particular, we aim to provide algorithms that make ART-type methods an attractive alternative to cone-beam FBP methods with regards to accuracy. Speed will be the topic of the next chapter. We will analyze the cone-beam reconstruction procedure of algebraic methods from a sampling theory point of view which lets us identify and eliminate the additional set of problems that algebraic algorithms have in the cone-beam setting. Most of these problems stem from the circumstance that in cone-beam the diverging rays sample and update the reconstruction grid in a non-uniform fashion, which leads to a considerable amount of aliasing-related noise artifacts for cone angles greater than 20˚. Even though these artifacts may have never been noticeable in high-contrast reconstructions, they become visible in the low-contrast case. We will then devise several strategies to eliminate these artifacts. In our analysis, we will mainly concentrate on ART and SART, as they offer a relatively different perspective of the reconstruction process, but are yet of similar efficiency.

The outline of this chapter is as follows. Section 4.1 moves ART into the cone beam setting, analyzes its shortcomings and presents solutions to overcome these deficiencies. Next, Section 4.2 discusses the use of SART for cone-beam reconstruction. Finally, Section 4.3 puts everything together and presents a variety of results obtained with our algebraic reconstruction testbed software. In this section, the effects of a wide range of ART parameters on both reconstruction quality and speed are investigated. The studied factors include: the value and functional variation of the relaxation coefficient λ, the relevance of volume initialization, and the effect of the algebraic correction algorithm (ART vs. SART).

4.1 Modified ART for Accurate Cone-Beam Reconstruction

In this section, we investigate the accuracy of ART in the context of low-contrast 3D cone-

beam reconstruction. We will find that ART in its present form is unsuitable in the cone-beam setting as it produces reconstructions with significant reconstruction artifacts. Henceforth, we will prescribe a number of modifications of ART's projection and backprojection mechanisms with which accurate reconstructions can be obtained, and which do not compromise the efficiency of ART. For quality assessment we will use a 3D extension of the Shepp-Logan brain phantom [61], similar to the one due to Axelsson [3]. The definition of our phantom is given in Table 4.1, while two orthogonal slices across the phantom are shown in Figure 4.1. From this phantom, we analytically compute 80 projection images of 128×128 pixels each, forcing equation (2.1) to be slightly underdetermined. The projections are obtained at equidistant angles φ within a range of $[0, 180^\circ+\gamma]$, where $\gamma/2$ is the cone half-angle.

ellipsoid	Center (mm)			Half-axis (mm)			Angle (degrees)		dens.
	x	y	z	x	y	z	θ	ϕ	
1	0.0	0.0	0.0	69.0	90.0	92.0	0.0	0.0	2.0
2	0.0	0.0	-1.84	66.24	88.0	87.4	0.0	0.0	-0.98
3	-22.0	-25.0	0.0	41.0	21.0	16.0	72.0	0.0	-0.02
4	22.0	-25.0	0.0	31.0	22.0	11.0	-72.0	0.0	-0.02
5	0.0	-25.0	35.0	21.0	35.0	25.0	0.0	0.0	0.01
6	0.0	-25.0	10.0	4.6	4.6	4.6	0.0	0.0	0.01
7	-8.0	-25.0	-60.5	4.6	2.0	2.3	0.0	0.0	0.01
8	6.0	-25.0	-60.5	4.6	2.0	2.3	90.0	0.0	0.01
9	6.0	6.25	-10.5	5.6	10.0	4.0	90.0	0.0	0.02
10	0.0	62.5	10.0	5.6	10.0	5.6	0.0	0.0	-0.02
11	0.0	-25.0	-10.0	4.6	4.6	4.6	0.0	0.0	0.01
12	0.0	-25.0	-60.5	2.3	2.3	2.3	0.0	0.0	0.01

TABLE 4.1 The definition of our 3D extension of the Shepp-Logan phantom [61], similar to the one used by [3]. The angles θ and ϕ are the polar and azimuthal angles of the ellipsoid *z*-axis. The scanner rotates about the *y*-axis.

Although detailed quantitative results are postponed to Section 4.3, we would like to illustrate the material in this section by the use of some examples. These examples will assume

(a) (b)

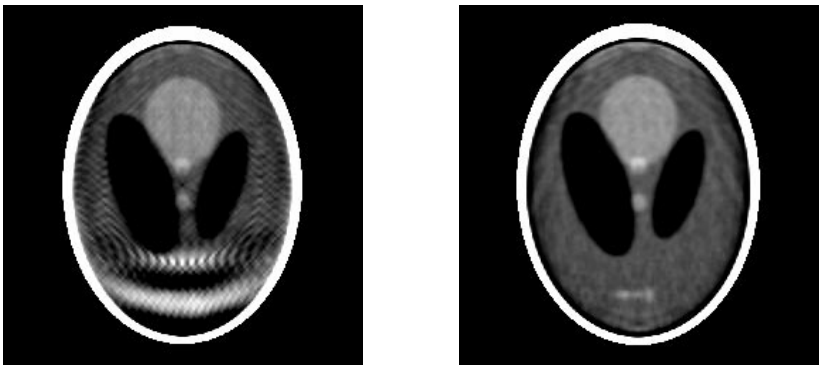
FIGURE 4.1: Slices across the 3D Shepp-Logan brain phantom: (a) y=-25mm, (b) z=8mm. The pixel intensities in these and all other slice images presented in this paper were thresholded to the interval [1.0,1.04] to improve the displayed contrast of the brain features. For example, the contrast between the three small tumors in the lower portion of the slice in (a) and the background matter is only 0.5% of the full density range.

certain settings of parameters such as λ, and $\boldsymbol{V}^{(0)}$, which will later be shown to be a good compromise between accuracy and speed of convergence.

4.1.1 Reconstruction artifacts in traditional cone-beam ART

Let us now use the ART algorithm of equation (2.11) to reconstruct a 128^3 volume from 80 projections with γ=60˚. λ is set to 0.08 and $\boldsymbol{V}^{(0)}=\mathbf{0}$. The reconstruction result of slice y=-25mm after 3 iterations is shown in Figure 4.2a. Here, we observe significant reconstruction artifacts which obliterate the small tumors in the lower image regions almost completely. For a comparison, Figure 4.2b shows a 3D reconstruction from parallel beam data with the same algorithm and parameter settings. No significant artifacts are present in that case.

Thus the artifacts must result from the cone-beam configuration in which rays emanate from the source and traverse the volume in a diverging fashion before finally hitting the projection plane. It may be suspected that it is this diverging nature of the rays that causes the reconstruction artifacts in Figure 4.2a. And indeed, more evidence is provided by Figure 4.3, where we show the reconstruction volume of a solid sphere (diameter=0.75n) after the first correction image (at φ=0˚) was applied, for both 60˚ cone-beam data (Figure 4.3a) and parallel-beam data (Figure 4.3b). In these figures, we choose the z-axis to coincide with the beam direction and z_c to be the location of the volume center slice perpendicular to the beam direction (see also Figure 4.4). In the cone-beam correction of Figure 4.3a, the non-uniform density distribution in the volume center slice along the cone direction (Figure 4.3a, side view) can be easily noticed. We see that much more energy is deposited in the volume slices close to the source where the ray density is high (Figure 4.3a, near slice,



(a) (b)

FIGURE 4.2: (a) The slice of Figure 4.1a reconstructed with traditional ART from cone-beam projection data (λ=0.08, γ=60˚, $\boldsymbol{V^{(0)}=0}$, 3 iterations). Notice the significant stripe artifacts that completely obliterate the small tumors. (b) A reconstruction of the same slice from parallel beam data using the same algorithm and parameter settings. This reconstruction does not have the strong artifacts of (a).

$z=z_n=z_c-0.25n$), while only little energy is deposited in the volume slices further away from the source where the ray density is low (Figure 4.3a, far slice, $z=z_f=z_c+0.25n$). In particular, the far slice displays a grid-like pattern which indicates an undersampling of the volume by the rays in this slice. This inadequate ray sampling rate potentially gives rise to aliasing, which is very likely to have caused the reconstruction artifacts of Figure 4.2a. The effects of aliasing are amplified since in ART the volume is projected and updated continuously, with every projection introducing additional aliasing into the reconstruction.

In contrast to the cone-beam case, the parallel-beam correction, shown in Figure 4.3b, provides a homogeneous density distribution. No excess density is deposited in the near slice at $z=z_n$ and no aliasing-prone, grid-like pattern is generated in the far slice at $z=z_f$. Thus reconstruction artifacts are unlikely to occur and, indeed, have not been observed in Figure 4.2b. In the following section, we will now investigate our observations more formally.

4.1.2 New scheme for projection/backprojection to prevent reconstruction artifacts

Both the projection and the backprojection algorithms must be adapted to avoid the aliasing problems outlined in the previous section. These enhancements make it necessary to modify ART's basic correction algorithm. We now describe these new concepts in more detail.

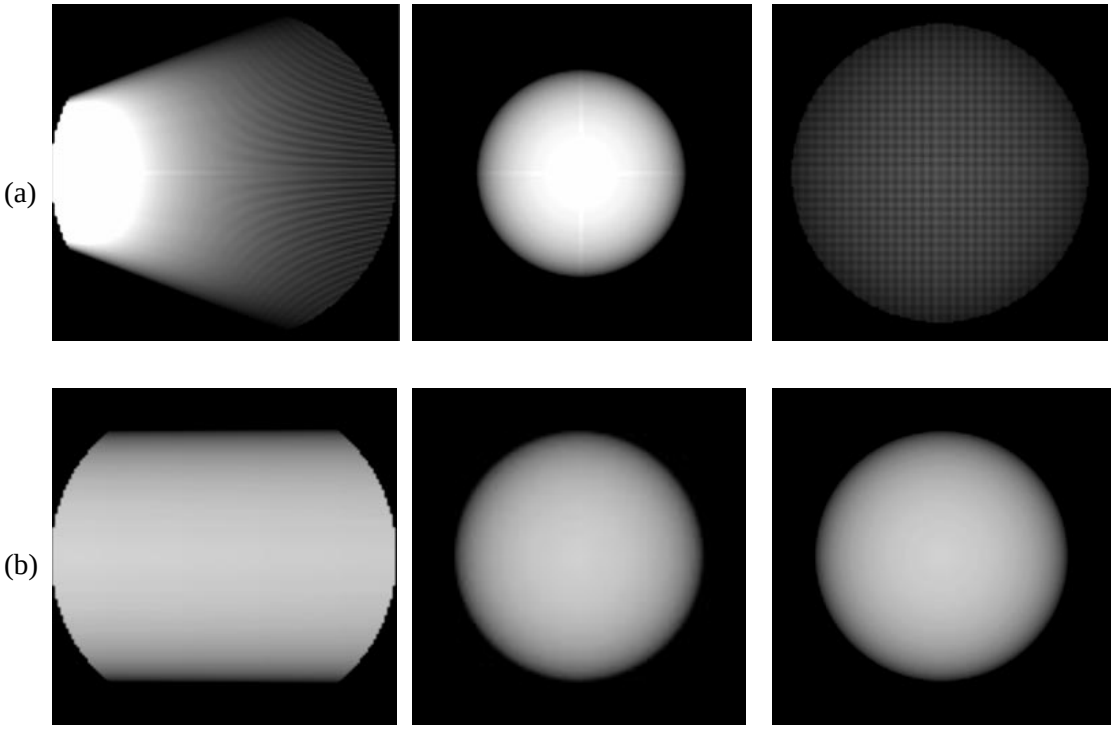


FIGURE 4.3: Reconstruction of a solid sphere (diameter=0.75n) after one correction at φ=0˚ was applied: (a) cone angle γ=60˚, (b) parallel beam, γ=0˚. The side view shows a cut across the center of the volume along z, e.g. the direction of the cone beam. With $z=z_c$ being the location of the volume center slice perpendicular to the cone direction (see also Figure 4.4), the near slice is the volume slice at $z=z_n=z_c-0.25n$ and the far slice is the volume slice at $z=z_f=z_c+0.25n$. Notice the uneven density distribution for the cone-beam reconstruction, while for parallel-beam the density is uniformly distributed.

4.1.2.1 Adapting the projection algorithm for cone-beam ART

In the usual implementation of ART, a pixel value $p_i^{(k)}$ is computed by the ray integral:

$$p_i^{(k)} = \sum_{i=1}^{N} v_j w_{ij} = \sum_{i=0}^{N} v_j \bar{h}(r_i) \tag{4.1}$$

where r_i is the ray going from the source to image pixel i, and $\bar{h}$ is the interpolation kernel function h pre-integrated in the direction of ray r_i. The ART weight factor w_{ij} that determines the amount of influence of voxel v_j on the pixel sum $p_i^{(k)}$ is thus given by $\bar{h}(r_i)$.

Although in ART a volume is updated for each ray separately, it is convenient for our discussion to treat all rays that belong to one projection image as an ensemble and act as if grid correction is performed only after all image rays have completed their forward projection. Doing so allows us to use principles from sampling theory to explain and subsequently eliminate the reconstruction artifacts observed before. We admit that this approach is slightly incorrect since in ART the projection sum of a ray belonging to a particular projection image always contains the grid corrections performed by the previous ray(s) of the same projection image. Due to this circumstance the projections and corrections obtained with ray-based ART and image-based ART are not strictly the same. However, this simplification has only a minor effect on the correctness of our analysis.

Consider now Figure 4.4a where the 2D case is illustrated. Here, the dashed lines denote the linear rays along which the volume is integrated. The rays that emanate from the source traverse the volume in form of a curved rayfront. Within this curved rayfront the rate $\omega_r=1/T_r$ at which the ensemble of rays samples the grid is constant (see Section 4.4). The further the rayfront is away from the source, the smaller is the ray ensemble's grid sampling rate. If one characterizes the position of the rayfront by the closest distance from the source, $s(z)$, then there is a $s(z)=z=z_c$ at which the rayfront samples the grid at exactly the grid sampling rate ω_g, e.g., $\omega_r=\omega_g=1/T_g$. Then, for $z<z_c$, the rayfront sampling rate is higher than the grid sampling rate, while for $z>z_c$, the rayfront sampling rate is lower than the grid sampling rate. For our discussion, we approximate the curved rayfronts by planar rayfronts or slices (see Section 4.4 for an error analysis). Thus, in Figure 4.4a, z_n is the location of the near slice of Figure 4.3 and z_f is the location of the far slice.

We mentioned earlier that by placing an interpolation kernel h at each grid voxel j and scaling it by the grid voxel's value v_j, we obtain a field of overlapping interpolation kernels that reconstructs the discrete grid function f_s into a continuous function f. Let us now decompose the volume into an infinite set of parallel slices along z. The contribution of a voxel j to the function $g(z)$ represented by a slice is then given by the 2D intersection of its interpolation kernel and the slice, denoted $h(z)$, while the sum of all scaled kernel intersections $h(z)$ produces the continuous slice function $g(z)$. A ray integral for pixel value $p_i^{(k)}$ is then computed by sampling all slices in the ray direction along z, which changes equation (4.1) into:

$$p_i^{(k)} = \int_z \sum_{j=1}^{N} v_j w_{ij}(z) = \int_z \sum_{j=1}^{N} v_j h(z, r_i) \tag{4.2}$$

The rayfront as a whole produces a sampled slice image $g_s(z)$ at each depth z (see Figure 4.4b). Hence, a complete projection image can be formed by adding these sampled slice images $g_s(z)$. This leads to an alternate expression for $p_i^{(k)}$:

$$p_i^{(k)} = \int_z g_s(z, iT_r) \tag{4.3}$$

Figure 4.4b illustrates that the ray sampling rate $1/T_r$ within each sampled slice image is not constant but is a linear function of z.

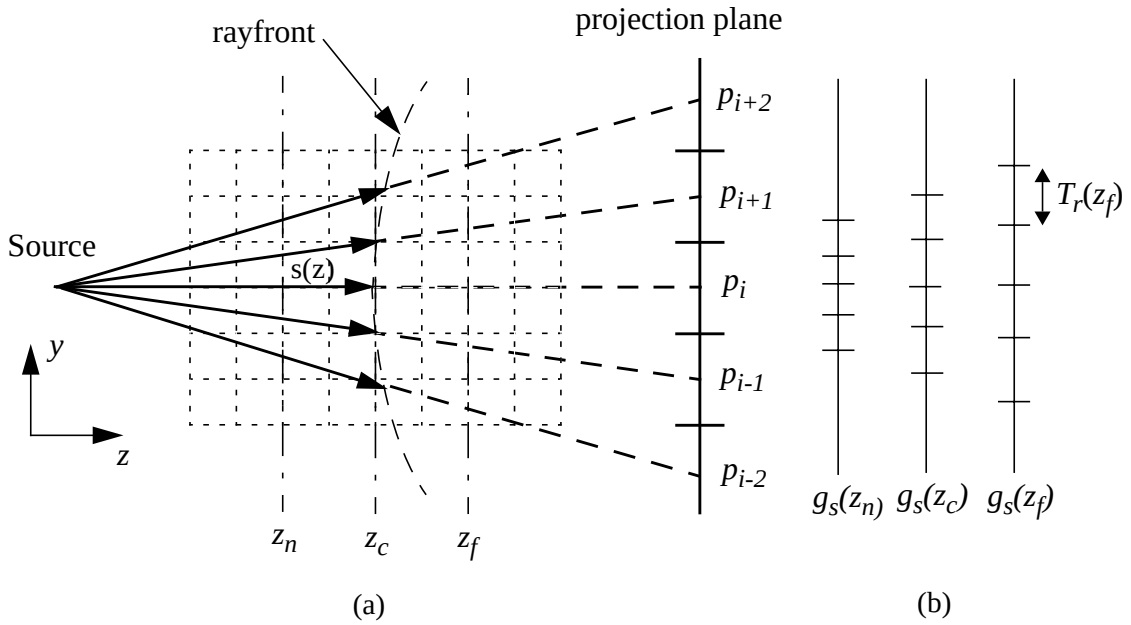


FIGURE 4.4: Perspective projection for the 2D case. (a) Rays emanate from the source into the volume along a curved rayfront which position is quantified by $s(z)$, the closest distance of the rayfront to the source. For our discussion, we approximate the curved rayfronts by planar rayfronts or slices. Then, z_c is the location of the volume center slice, z_n is the location of the near slice of Figure 4.3a and z_f is the location of the far slice of Figure 4.3a. (b) Slice images $g_s(z)$ for $z=z_n$, z_c, and z_f. The sampling period T_r in each of the slice images is determined by the distance z of the slice image from the source: $T_r(z_n)<T_r(z_c)<T_r(z_f)$.

The process in which an ensemble of rays in a rayfront at depth z generates a sampled slice image $g_s(z)$ can be decomposed into two steps:

1. Reconstruction of the discrete grid signal $f_s(z)$ into a continuous signal $f(z)$ by convolving $f_s(z)$ with the interpolation filter h.
2. Sampling $f(z)$ by a comb function with period $T_r=T_g z/z_c$.

This can be written as:

$$\begin{aligned} g_s(z, iT_r) &= comb\left(\frac{y}{T_r}\right) \cdot \left(f_s(z, kT_g) * h\left(\frac{y}{T_g}\right)\right) \\ &= comb\left(\frac{y}{T_g z/z_c}\right) \cdot \left(f_s(z, kT_g) * h\left(\frac{y}{T_g}\right)\right) \end{aligned} \tag{4.4}$$

Here, and in all following equations, $k \in \aleph$.

In the frequency domain, equation (4.4) is expressed as follows:

$$\begin{aligned} G_s(z,\nu) &= \frac{1}{\omega_g z_c / z} comb\left(\frac{\nu}{\omega_g z_c / z}\right) * \left(\sum_{-\infty}^{\infty} F(\nu - k\cdot\omega_g)\cdot H\left(\frac{\nu}{\omega_g}\right)\right) \\ &= \sum_{-\infty}^{\infty} F(\nu - k\cdot\omega_g z_c / z) = \sum_{-\infty}^{\infty} F(\nu - k\cdot\omega_r) \end{aligned} \tag{4.5}$$

using the relationship

$$X\cdot comb(\nu X) = \sum_{i=\infty}^{\infty} \delta\left(\nu - \frac{k}{X}\right) \tag{4.6}$$

If the backprojection is performed correctly, we must assume that the grid contains frequencies of up to but not greater than $\omega_g/2$. Then, since h has bandwidth ω_g and subsequently $f=h*f_s$ has also bandwidth ω_g, there is a chance that the signal aliases of G_s overlap in the frequency domain whenever $\omega_r<\omega_g$, i.e., $z>z_c$ (see Figure 4.6a and b). Thus each slice with $z>z_c$ potentially contributes an aliased signal to the composite projection image.

We can fix this problem by adding a lowpass filter L_P between H and the sampling process for all slices with $z>z_c$:

$$\begin{aligned} G_s(z,\nu) &= \frac{1}{\omega_g z_c / z} comb\left(\frac{\nu}{\omega_g z_c / z}\right) \\ &\quad * \left(\sum_{-\infty}^{\infty} F(\nu - k\cdot\omega_g)\cdot H\left(\frac{\nu}{\omega_g}\right)\cdot L_P\left(\frac{\nu}{\omega_g z_c / z}\right)\right) \qquad z > z_c \end{aligned} \tag{4.7}$$

An efficient way of implementing this concept is to pre-convolve H with a lowpass filter L_p, say a boxfilter B of width z/z_c, and use this new filter HB in place of H for interpolation. An alternative method is to simply decrease the bandwidth of H to $\omega_g z_c/z$. which gives rise to a filter H':

$$\begin{aligned} G_s(z,\nu) &= \frac{1}{\omega_g z_c / z} comb\left(\frac{\nu}{\omega_g z_c / z}\right) \\ &\quad * \left(\sum_{-\infty}^{\infty} F(\nu - k\cdot\omega_g)\cdot H\left(\frac{\nu}{\omega_g z_c / z}\right)\right) \qquad z > z_c \end{aligned} \tag{4.8}$$

This technique was also used in [62] to achieve accurate perspective volume rendering with the splatting technique. The frequency response of H' is shown in Figure 4.5a for the slice at $z=z_f$ where $z_f / z_c = 1.42 T_g$. The frequency response of HB is also shown. We notice that the frequency responses of both filters are similar and we also observe that both effectively limit the bandwidth to $\omega_r=0.7\omega_g$. Note, however, that although reducing the bandwidth of

h for $z>z_c$ properly separates the signal aliases, the upper frequency bands of the grid signal in these regions have been filtered out by the lowpass operation and will not contribute to the projection images (see Figure 4.6c).

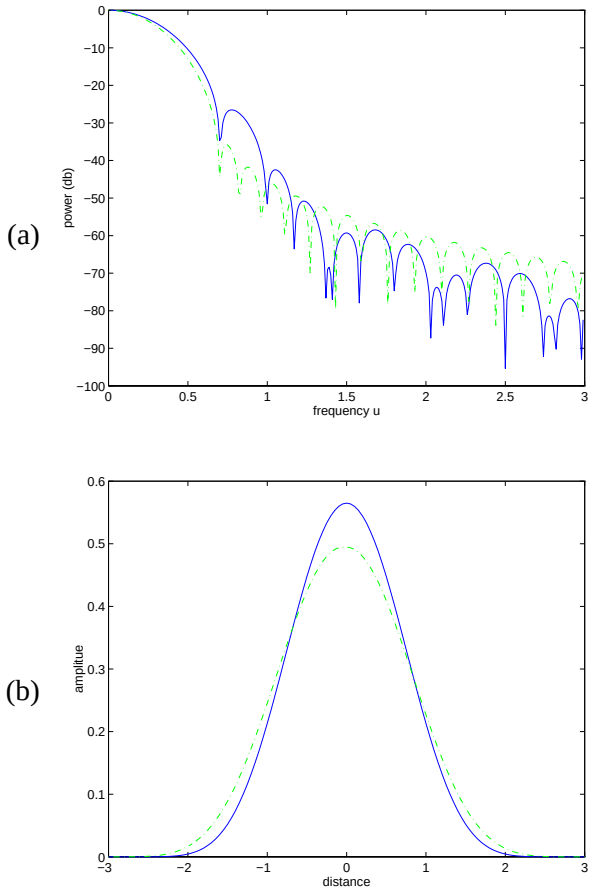


FIGURE 4.5: (a) Frequency response of the interpolation filter *H'* and the combined filter *HB*, at $z_f/z_c = 1.42T_g$, (b) Impulse response of the same filters *h'* and *hb* at $z/z_c=1.42T_g$ (dash-dot: *h'*, solid: *hb*).

According to the Fourier scaling property, *h'* is obtained from *h* by stretching and attenuating it in the spatial domain. The sampled slice signal is then:

$$g_s(z, iT_r) = comb\left(\frac{y}{T_r}\right) \cdot \left(f_s(z, kT_g) * \frac{1}{z/z_c} h\left(\frac{y}{T_g z/z_c}\right)\right) \qquad z > z_c \tag{4.9}$$

This filter *h'* is shown in Figure 4.5b and is also contrasted with *hb*. The spatial extent of *hb* is $|hb|=|h|+|b|$, while the spatial extent of *h'* is $|h'|=|b||h|=z/z_c|h|$ (for $T_g=1.0$). Thus for $|b|<|h|/(|h|-1)$, $|h'|<|hb|$. Therefore, if $|h|=4.0$, then as long as $z/z_c<1.33$, *h'* is more efficient to use for interpolation. Since the majority of all interpolations occur in this range, the use

of h' is to be preferred.

Finally, although one must use a stretched and attenuated version of h to lowpass f before sampling it into g_s when $z>z_c$, one cannot use a narrow and magnified version of h when $z<z_c$. Doing so would increase h's bandwidth above ω_g and would introduce higher order aliases of f_s into the reconstructed f (see Figure 4.6d). Hence, we must use h in its original width when $z<z_c$ (Figure 4.6e).

Not only the ART projection step is affected by the non-uniform grid sampling rate of the cone-beam rays, the backprojection step must also be adapted. This is discussed next.

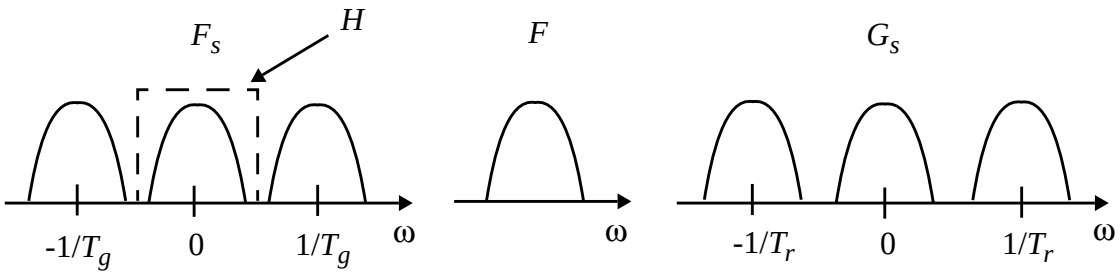
4.1.2.2 Adapting the backprojection algorithm for cone-beam ART

In backprojection, just like in forward projection, the volume is traversed by an ensemble of divergent rays. However, in backprojection the rays do not traverse the volume to gather densities, instead they deposit (corrective) density into the volume. In other words, the volume now samples and interpolates the ensemble of (correction) rays instead of the rays sampling and interpolating the volume. For the following discussion recall that, for convenience, we assume that all N_r rays of a projection P_φ first complete their projection and then all simultaneously correct the volume voxels. Let us now again decompose the volume into an infinite set of slices along the z axis, oriented perpendicular to the beam direction, and consider the corrections for the voxels within each slice separately. Each ray r_i carries with it a correction factor $corr_i$, computed by the fraction in equation (2.11). Like in the projection phase, we use $h(z)$ as the interpolation filter within a slice. Then the total correction dv_j to update v_j is given by the sum of all ray corrections $corr_i$, $1 \le i \le N_r$, for voxel j within a slice, integrated over all slices. This gives rise to the following equation which is similar to equation (4.2):

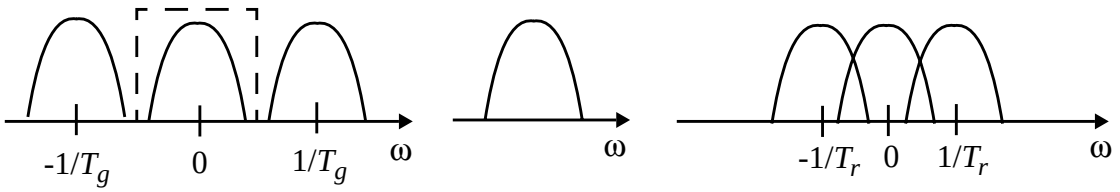
$$dv_j = \int_z \sum_{i=1}^{N_r} corr_i \cdot w_{ij}(z) = \int_z \sum_{i=1}^{N_r} corr_i \cdot h(z, r_i) \tag{4.10}$$

This equation is in line with the traditional viewpoint of ART. Let us now consider an alternative representation that will help us to explain the aliasing artifacts of cone-beam. We observe from Figure 4.4 that the intersection of the ensemble of rays with a slice, say the one at $z=z_f$, gives rise to a discrete image $c_s(z)$ with the pixel values being the correction factors $corr_i$. Note that the pixel rate in the $c_s(z)$ is a linear function of z, which is equivalent to the situation for the $g_s(z)$ in the projection case (illustrated in Figure 4.4b). In order to compute the correction factors dv_j, the voxel grid then samples and interpolates each of the slice images $c_s(z)$ into a correction image $d_s(z)$ with constant pixel period T_g, and each voxel j integrates all its correction factors in the $d_s(z)$ along z. Again, we use $h(z)$ as the interpolation filter within a slice, only now it is indexed by the discrete voxel locations and not by the traversing rays. The correction dv_j for voxel j can then be expressed as follows:

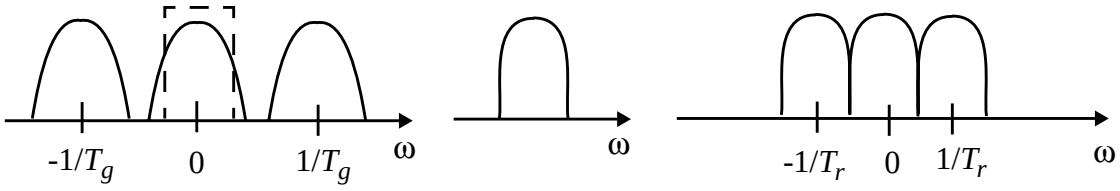
$$dv_j = \int_z \sum_{i=1}^{N_r} c_s(z, iT_r)\, h(z, v_j) = \int_z d_s(z, jT_g) \tag{4.11}$$



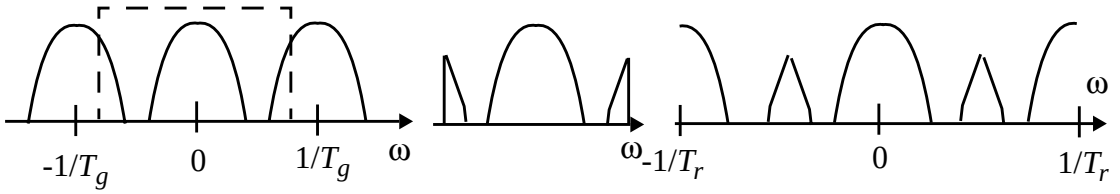
(a) $z = z_c$, Bandwidth(H) = $1/T_g = 1/T_r$



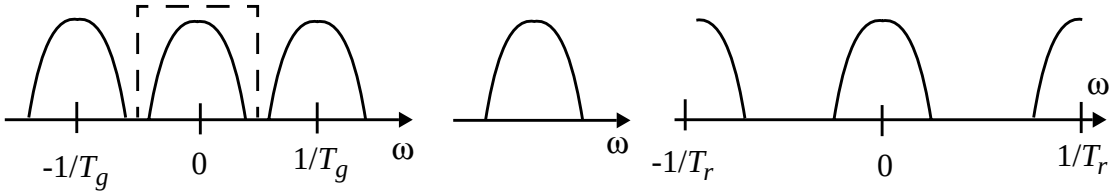
(b) $z > z_c$, Bandwidth(H) = $1/T_g$



(c) $z > z_c$, Bandwidth(H) = $1/T_r$



(d) $z < z_c$, Bandwidth(H) = $1/T_r$



(e) $z < z_c$, Bandwidth(H) = $1/T_g$

FIGURE 4.6: Grid projection at different slice depths in the frequency domain. Shown in the left column is the frequency spectrum of the sampled grid signal, F_s, with the filter H superimposed. In the center column is the reconstructed signal F, while in the right column is the reconstructed F resampled into the slice image G_s.

Let us concentrate on one interpolated correction image $d_s(z)$. Since the interpolated signal is now the ensemble of rays and the sampling process is now the volume grid, the roles of ω_r and ω_g in equations (4.4) and (4.5) are reversed. The interpolation of the rayfront slice image $c_s(z)$ into a correction image $d_s(z)$ by the voxel grid can be decomposed into two steps:

1. Reconstruction of the discrete correction rayfront signal $c_s(z)$ into a continuous signal $c(z)$ by convolving $c_s(z)$ with the interpolation filter h. Notice that, for now, in order to capture the whole frequency content of c_s, we set the bandwidth of h to the bandwidth of the rayfront grid, $1/T_r = z_c/zT_g$. This is a new approach to ART, as usually the bandwidth of h is always set to ω_g, along with an amplitude scale factor of 1.0.
2. Sampling $c(z)$ into $d_s(z)$ by a comb function with period T_g.

This is formally written as:

$$\begin{aligned} d_s(z, jT_g) &= comb\left(\frac{y}{T_g}\right) \cdot \left(c_s(z, kT_r) * h\left(\frac{y}{T_r}\right)\right) \\ &= comb\left(\frac{y}{T_g}\right) \cdot \left(c_s(z, kT_g z/z_c) * h\left(\frac{y}{T_g z/z_c}\right)\right) \end{aligned} \tag{4.12}$$

In the frequency domain, this is expressed as follows:

$$\begin{aligned} D_s(z, v) &= \frac{1}{\omega_g} comb\left(\frac{v}{\omega_g}\right) * \left(\sum_{-\infty}^{\infty} C(v - k \cdot \omega_r) \cdot H\left(\frac{v}{\omega_r}\right)\right) \\ &= \sum_{-\infty}^{\infty} C(v - k \cdot \omega_g) = \sum_{-\infty}^{\infty} C(v - k \cdot \omega_r z/z_c) \end{aligned} \tag{4.13}$$

again using the relationship of equation (4.6). Since C's bandwidth is always ω_r, constrained by H, we get overlapping aliases in D_s when $z<z_c$ (see Figure 4.6b). However, when $z \geq z_c$, no overlap occurs (Figure 4.6a). This means that only for $z<z_c$ we need to limit H to the bandwidth of the grid, ω_g, resulting in a filter H' (Figure 4.6c). For all other z, we use the bandwidth of the rayfront ω_r. More formally:

$$\begin{aligned} D_s(z, v) &= \frac{1}{\omega_g} comb\left(\frac{v}{\omega_g}\right) * \left(\sum_{-\infty}^{\infty} C(v - k \cdot \omega_r) \cdot H\left(\frac{v}{\omega_g}\right)\right) \qquad z < z_c \\ &= \frac{1}{\omega_g} comb\left(\frac{v}{\omega_g}\right) * \left(\sum_{-\infty}^{\infty} C(v - k \cdot \omega_r) \cdot H\left(\frac{v}{\omega_r z/z_c}\right)\right) \quad z < z_c \end{aligned} \tag{4.14}$$

Again, according to the Fourier scaling property, *h*' is obtained from *h* by stretching and attenuating it in the spatial domain. The sampled slice signal is then:

$$\begin{aligned} d_s(z, jT_g) &= comb\left(\frac{y}{T_g}\right) \cdot \left(c_s(z, kT_r) * \frac{1}{z_c/z} h\left(\frac{y}{T_r z_c/z}\right)\right) \qquad z < z_c \\ &= comb\left(\frac{y}{T_g}\right) \cdot \left(c_s(z, kT_r) * \frac{1}{z_c/z} h\left(\frac{y}{T_g}\right)\right) \qquad z < z_c \end{aligned} \tag{4.15}$$

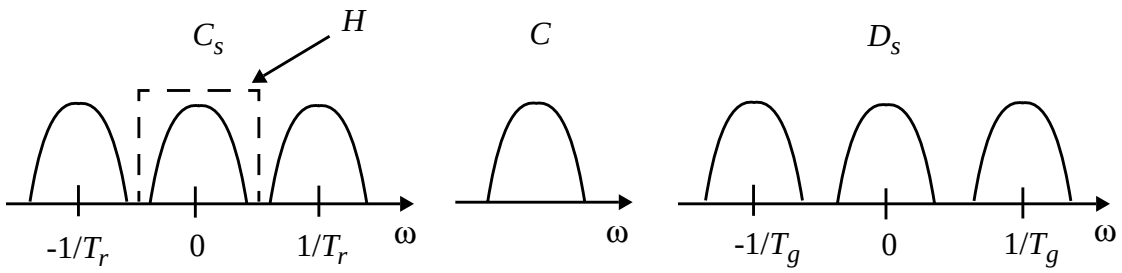
Thus for $z<z_c$ the bandwidth of *H*' is ω_g and the scaling factor in the spatial domain is z/z_c. Reducing the amplitude of *h* when $z<z_c$ prevents the over-exposure of corrective density in the volume slices near the source (as evident in Figure 4.3a). On the other hand, for $z>z_c$, the bandwidth of *H*' is $\omega_r=\omega_g z_c/z$ and the scaling factor in the spatial domain is 1.0. Increasing the width of *h* when $z>z_c$ prevents the under-exposure of some voxels in the volume slices far from the source and facilitates a homogenous distribution of the corrective energy. If we used a more narrow filter for $z>z_c$ then we would introduce the higher order aliases of $c_s(z)$ into the reconstructed $c(z)$, which is manifested by the aliased grid-like pattern in the far-slice of Figure 4.3a (see also Figure 4.6d and e).

Note that if projection and backprojection are performed with the proper filtering as described here, then the reconstruction volume will never contain frequencies greater than ω_g. Thus our previous argument that the grid projection may assume that the volume frequencies never exceed ω_g is correct. Of course, there is a small amount of aliasing introduced by the imperfect interpolation filter, but these effects can be regarded negligeable as the Bessel-Kaiser filter has a rather fast decay in its stop-band (see Figure 4.5a).

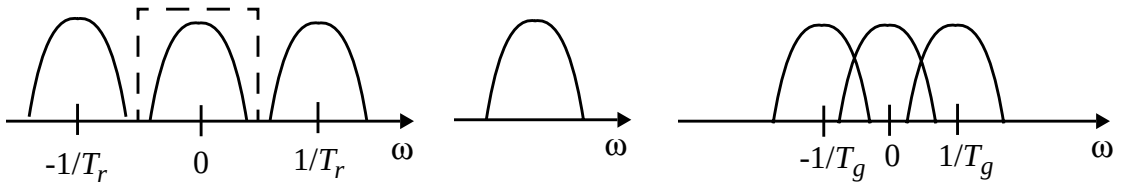
4.1.2.3 Putting everything together

Since we are using an interpolation kernel that is pre-integrated into a 2D footprint table we cannot represent the depth dependent kernel size accurately and still use the same footprint everywhere in the volume. An accurate implementation would have to first distort the 3D kernel with respect to the depth coordinate *z* and then integrate it along *z*. Since the amount of distortion changes for every *z*, we would have to do this process for every voxel anew. Since this is rather inefficient, we use an approximation that utilizes the generic footprint table of an undistorted kernel function and simply stretches it by z_v/z_c whenever appropriate. Here, z_v is the depth coordinate of the voxel center. Thus the volume regions covered by the voxel kernel for which $z<z_v$ are lowpassed too much, while for $z>z_v$ the amount of lowpassing is less than required. However, since the kernel extent is rather small and the volume bandlimit rarely reaches the Nyquist limit, this error is not significant.

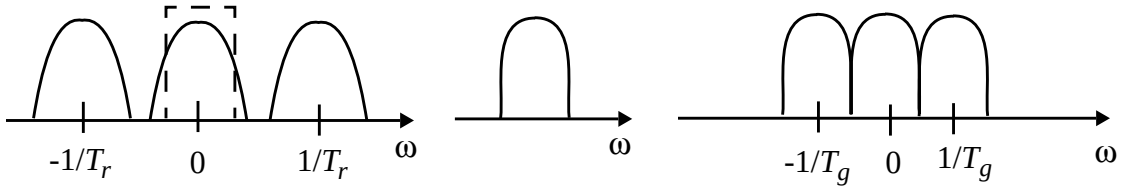
Another issue is how one goes about estimating the local sampling rate ω_r of the grid of rays. Highest accuracy is obtained when the curved representation of the rayfronts is used (see Figure 4.4). In this case one would compute the normal distance between the set of neighboring rays traversing the voxel for which the kernel needs to be stretched (see also Section 4.4). An easier way that is nevertheless a good approximation is to just use the



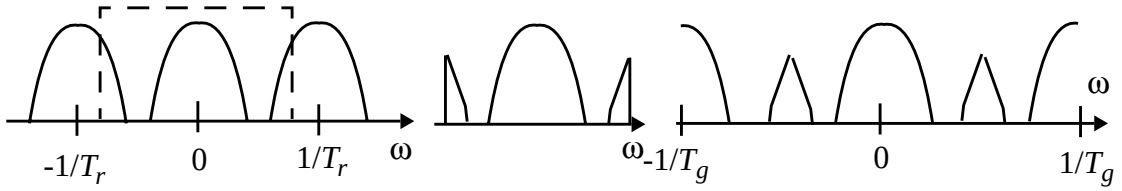
(a) $z = z_c$, Bandwidth(H) = $1/T_r = 1/T_g$



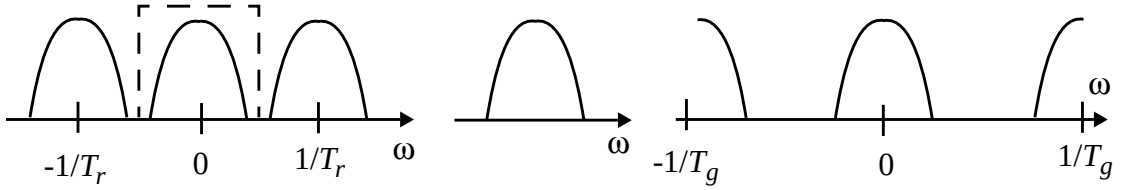
(b) $z < z_c$, Bandwidth(H) = $1/T_r$



(c) $z < z_c$, Bandwidth(H) = $1/T_g$



(d) $z > z_c$, Bandwidth(H) = $1/T_g$



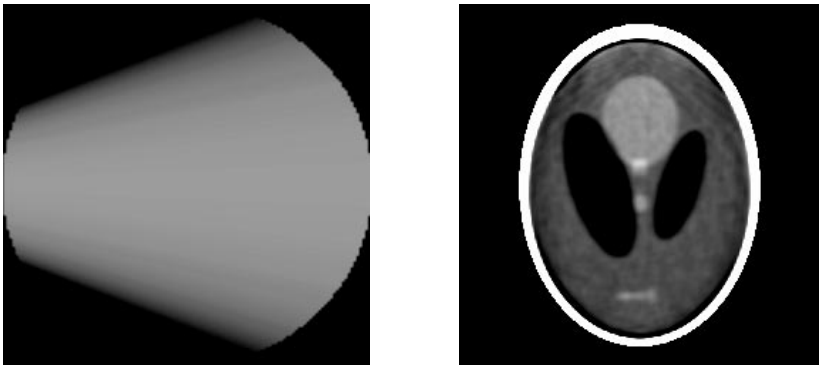
(e) $z > z_c$, Bandwidth(H) = $1/T_r$

FIGURE 4.7: Backprojection at different slice depths in the frequency domain. Shown in the left column is the frequency spectrum of the discrete correction signal, C_s, with the filter H superimposed. In the center column is the reconstructed correction signal C, while in the right column is the reconstructed C resampled into the grid slice image D_s.

voxel's *z*-coordinate in conjunction with the method of similar triangles to estimate ω_r (see Section 4.4 for a quantitative error analysis). This approximates the curved rayfronts to planar rayfronts, as we did in our theoretical treatment. Various fast algorithms for the estimation of ω_r are subject of the next chapter.

Since the interpolation kernel is stretched and scaled differently for projection and backprojection, the computed weights in these stages are different as well. Thus equation (2.11) changes to:

$$v_j^{(k+1)} = v_j^{(k)} + \lambda \frac{p_i - \sum_{n=1}^{N} w_{in}^F v_n^k}{\sum_{n=1}^{N} w_{in}^F w_{in}^B} w_{ij}^B \tag{4.16}$$



(a) (b)

FIGURE 4.8: Reconstruction with ART utilizing the new variable-width interpolation kernels: (a) The cone-cut of Figure 4.3a after one projection was processed: we observe a uniform distribution of the correction factors. (b) The slice of Figure 4.1a (λ=0.08, γ=60˚, $\boldsymbol{V^{(0)}=0}$, 3 iterations): the cone-beam reconstruction artifacts apparent in Figure 4.2a were successfully eliminated by using the new filter.

Here, w_{in}^F is the weight factor used for the forward projection and w_{in}^B is the weight factor used for backward projection.

Figure 4.8 shows images of a reconstruction obtained with the same parameters settings than Figure 4.2a, but using the new approach for cone-beam ART. Figure 4.8a shows the cone-cut of Figure 4.3a after one projection was processed. As was to be expected, the correction cone has now uniform intensity. Finally, Figure 4.8b shows the reconstructed slice of Figure 4.1a. Here we observe that this reconstruction no longer displays the strong aliasing artifacts that dominated Figure 4.2a.

4.2 3D Cone-Beam Reconstruction with SART

Let us now investigate SART and its behavior in the cone-beam setting. The bracketed part in the numerator of equation (2.17) is equivalent to the numerator of equation (2.11). Hence, the projection process of SART is identical to the one of ART and Section 4.1.2.1 applies again. However, SART's backprojection process differs from the one of ART. In contrast to ART, in SART, after the ray grid has been interpolated by the volume grid, a voxel correction is first normalized by the sum of influential interpolation filter weights before it is added to the voxel. This changes equation (4.10) to:

$$dv_j = \frac{\int_z \sum_{i=1}^{N_r} corr_i \cdot w_{ij}(z)}{\int_z \sum_{i=1}^{N_r} w_{ij}(z)} = \frac{\int_z \sum_{i=1}^{N_r} corr_i \cdot h(z, r_i)}{\int_z \sum_{i=1}^{N_r} h(z, r_i)} = \frac{\sum_{i=1}^{N_r} corr_i \cdot h(r_i)}{\sum_{i=1}^{N_r} \bar{h}(r_i)} \qquad (4.17)$$

We will see that this provides for a smoothly interpolated correction signal, even when the bandwidth of the interpolation filter is a constant ω_g, as is the case in the traditional ART and SART approaches. Let us now illustrate this behavior by ways of an example, illustrated in Figure 4.9. Here, we utilize a uniform discrete correction signal (spikes, $c_s(z)$ in Section 4.1.2.2), a 1D version of the one used in Figure 4.3, and a linear interpolation filter h (dotted line) for ease of illustration. In this figure, we show the result of the signal reconstruction with h, i.e., the signal $c(z)$ *prior* to sampling into $d_s(z)$ by the voxel grid. Consider now Figure 4.9a, where the situation for $z<z_c$ is depicted. In this case, the uniform correction signal has a rate $\omega_r>\omega_g$. In accordance to our previous results, we see that if $c_s(z)$ is reconstructed with traditional ART utilizing an h with constant bandwidth ω_g, we obtain a signal $c(z)_{\mathrm{ART}}$ with an amplitude much higher than the original correction signal. We know that we can avoid this problem by reducing the interpolation filter magnitude, which gives rise to the correctly scaled signal $c(z)_{ART}^{\text{var. kern.}}$. The correction signal reconstructed with SART, $c(z)_{\mathrm{SART}}$, on the other hand, does not require a reduction of the interpolation filter amplitude, since for each voxel the sum of interpolated ray corrections is normalized by the sum of the respective interpolation filter weights. Thus, with SART the reconstructed correction signal has always the correct amplitude. For $z>z_c$ (see Figure 4.9b) when $\omega_r<\omega_g$, we observe an aliased signal of half the grid frequency for $c(z)_{\mathrm{ART}}$ instead of the expected constant signal. Sampling $c(z)_{\mathrm{ART}}$ into $d_s(z)$ would then result in a grid-like pattern, similar to the one observed in the far slice of Figure 4.3a. We know, however, from the previous discussion that we can eliminate this effect by widening h with respect to ω_r. This yields the correct signal $c(z)_{\mathrm{ART}}^{\text{var. kern.}}$. For SART, on the other hand, a smooth and uniform $c(z)_{\mathrm{SART}}$ of correct amplitude is obtained even with the original interpolation filter width - again due

to the normalization step.

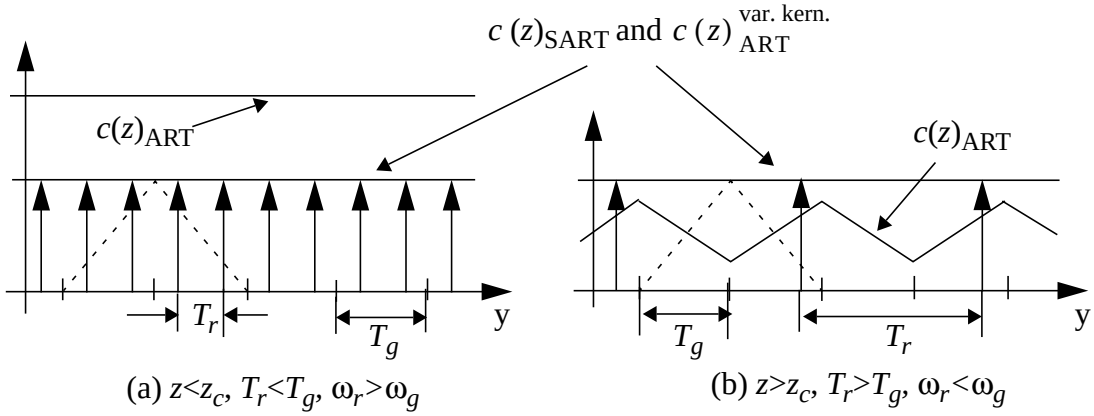


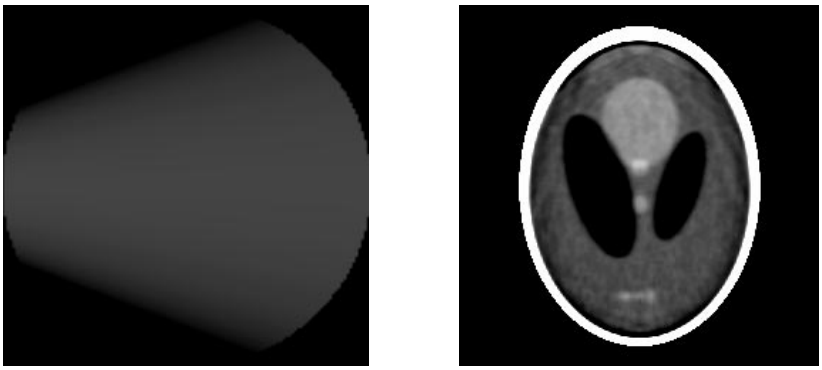
FIGURE 4.9: Reconstructing a uniform discrete correction signal $c_s(z)$ (spikes) into the continuous signal $c(z)$ with ART and SART using a linear interpolation filter h (depicted here for a bandwidth of ω_g in a dotted line). Signal $c(z)_{ART}$ is the reconstruction of $c_s(z)$ with ART and the traditional interpolation filter of bandwidth ω_g. In case (a), when $z<z_c$, the correction signal is magnified, as in Figure 4.3a, near slice. In case (b), when $z>z_c$, the correction signal is aliased, as in Figure 4.3b, far slice. As previously discussed, we can avoid these effects using variable interpolation kernels, i.e., by scaling the amplitude of h in case (a) and by widening h in case (b), which yields the correct signal $c(z)^{var.\ kern.}_{ART}$ in both cases. The normalization procedure of SART prevents magnification and aliasing artifacts and thus in both cases (a) and (b) a correctly reconstructed signal $c(z)_{SART}$ identical to $c(z)^{var.\ kern.}_{ART}$ is produced.

Consider now Figure 4.10a which shows the actual cone-cut of Figure 4.3a for SART. We see that the distribution of correction energy is homogeneous, even though the bandwidth of the interpolation filter was set to ω_g everywhere in the volume. Finally, Figure 4.10b shows a reconstruction obtained with SART under the same conditions than the previous ART reconstructions. We observe that no significant reconstruction artifacts are noticeable. Thus we can conclude that the SART approach is inherently more adequate for algebraic cone-beam reconstruction than the traditional, unmodified, ART approach.

4.3 Results

In order to assess the goodness of the reconstructions, we define two figures of merit: (i) the correlation coefficient CC between the original 3D Shepp-Logan brain phantom (SLP) and the reconstructed volume and (ii) the background coefficient of variation CV. These measures were also used by Ros et.al. [55].

We calculate CC within two selected regions of the SLP: (i) the entire inside of the head (without the skull) and (ii) an ellipsoidal volume containing the three small SLP tumors, as shown at the bottom of the brain section of Figure 4.1a and described by ellipsoids 7, 8, and



(a) (b)

FIGURE 4.10: Reconstruction with SART: (a) The cone-cut of Figure 4.3a, after one projection was processed. (b) The slice of Figure 4.1a (λ=0.3, γ=60˚, $\boldsymbol{V^{(0)}=0}$, 3 iterations). The aliasing artifacts apparent in Figure 4.2a do not exist.

12 in the SLP definition (see Table 4.1). While (i) measures the overall correlation of the reconstructed volume with the original, (ii) captures the sensitivity of the reconstruction algorithm to preserving small object detail. The CC is defined by:

$$CC = \frac{\sum_i (v_i - \mu_v)\,(o_i - \mu_o)}{\sqrt{\sum_i (v_i - \mu_v)^2 \sum_i (o_i - \mu_o)^2}} \tag{4.18}$$

where o_i and v_i are the values of the voxels in the original phantom volume and the reconstructed volume, respectively, and μ_o and μ_v are their means.

The background coefficient of variation CV is calculated within four ellipsoidal featureless regions located in different areas of the SLP. Since these regions are ideally uniform, CV represents a good measure for the noise present in the reconstructed SLP. The overall CV is the average of the four individual CVs and is defined as:

$$CV = \frac{1}{4}\sum_i^4 \frac{\sigma_i}{\mu_v} \tag{4.19}$$

where σ_i is the standard deviation of the voxel values within region *i*.

Using CV and CC to assess reconstruction quality, we have conducted a thorough study of the effects of various parameter settings:

- The initial state of the reconstruction volume: (i) zero, (ii) the average value of one of the projection images, properly scaled, and (iii) the average value of the SLP brain matter (e.g. 1.02). The last initialization method is hard to achieve in practice, since one usually does not know the average value of the reconstructed object in advance.

- The setting of the relaxation coefficient λ: experiments revealed that values in the range of [0.02, 0.5] yielded reconstructions that offered the best balance with respect to reconstruction quality and number of iterations. Therefore, reconstructions were obtained for λ=0.5, 0.3, 0.1, 0.08, 0.05, and 0.02.
- Time-weighted variation of λ during the first iteration: starting from 10% of the final value we have used: (i) immediate update to the final value, (ii) a linear increase and (iii) an increase due to a shifted cosine function (similar to [1]).
- The correction algorithm: (i) ART and (ii) SART.
- The size of the interpolation kernel: (i) constant and (ii) depth-dependent, as described in Section 4.1.2.
- The cone angle: 20˚, 40˚, and 60˚.
- The number of iterations: 1 through 10.

During our experiments we found that there is no significant difference in reconstruction quality whether λ is varied linearly or with a cosine function. Thus we only report results using a linear variation.

Figure 4.11 shows a number of plots illustrating CC and CV for various reconstruction methods and parameter settings. Figure 4.12 shows reconstructed SLP slice images after 3 iterations. The number of 3 iterations was chosen because the plots indicate that after this number of iterations both CC and CV have reached a level close to their final value. In both Figure 4.11 and Figure 4.12 the following terminology is used. Each parameter combination is described by a 3 digit code. The first digit codes the size of the interpolation kernel (C for constant, V for variable, depth-dependent). The second digit codes the volume initialization method (- for zero, I for projection average, I* for SLP average). Lastly, the third digit codes the function at which λ is varied during the first iteration (- for no variation, G for linear increase). Reconstructions were obtained for each combination of parameter settings over the length of 10 iterations for all six settings of λ. In order to keep the amount of presented data manageable, we do not plot the effect of λ in Figure 4.11. Instead, at every iteration and parameter combination we pick the lowest CV or highest CC, respectively, that was obtained in the set of six λ-dependent reconstructions. As the ideal λ is somewhat object-dependent anyway [25], this does not represent a serious trade-off. During the course of our experiments, we found that for ART and the SLP the λ that produces good reconstructions within four iterations is somewhere around 0.08, for SART this λ setting is around 0.3.

Figure 4.11a-c shows the CC for the SLP tumors at a cone angle of 40˚ for the three main correction methods used: ART using the constant interpolation kernel, ART using the variable-size interpolation kernel, and SART using both kernels. In Figure 4.11a we see that the reconstructions for G are always better than the respective ones without G. Since it is not any costlier to do G, the following plots will always use G. In Figure 4.11b we see that reconstruction results with VIG are similar to the ones with VI*G. Since VI*G is not realistic anyway, we will use only VIG for the remaining discussion. In the same figure we also observe that V-G, VI- and VI*- yield only marginally worse results than VIG, thus we will

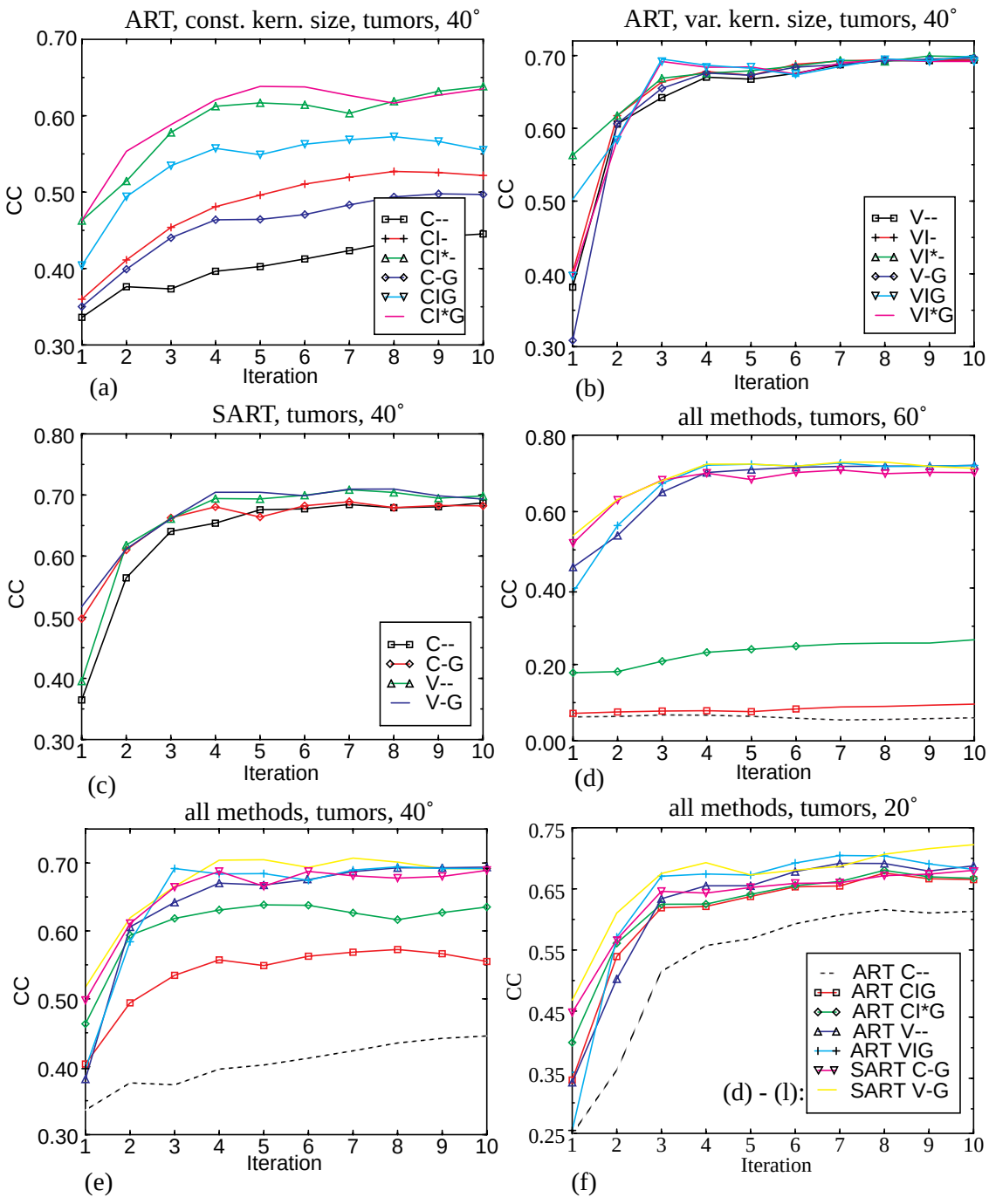


FIGURE 4.11: Correlation Coefficient (CC) and Background Coefficient of Variation (CV) for ART and SART with constant and variable interpolation kernel size for 3 regions of the SLP (u 80 projections of 128^2 pixels each). Each reconstruction was performed for λ=0.5, 0.3, 0.1, 0.08, 0.05, and 0.02, the graphs plot the highest CC/lowest CV of all λ settings at each iteration. (C: constant sized kernel, V: variable sized kernel, I: volume initialized to average of projection 0, I*: volume initialized to average of the SLP, G: linear increase of λ during the first iteration, -: volume initialization to zero or constant λ.)

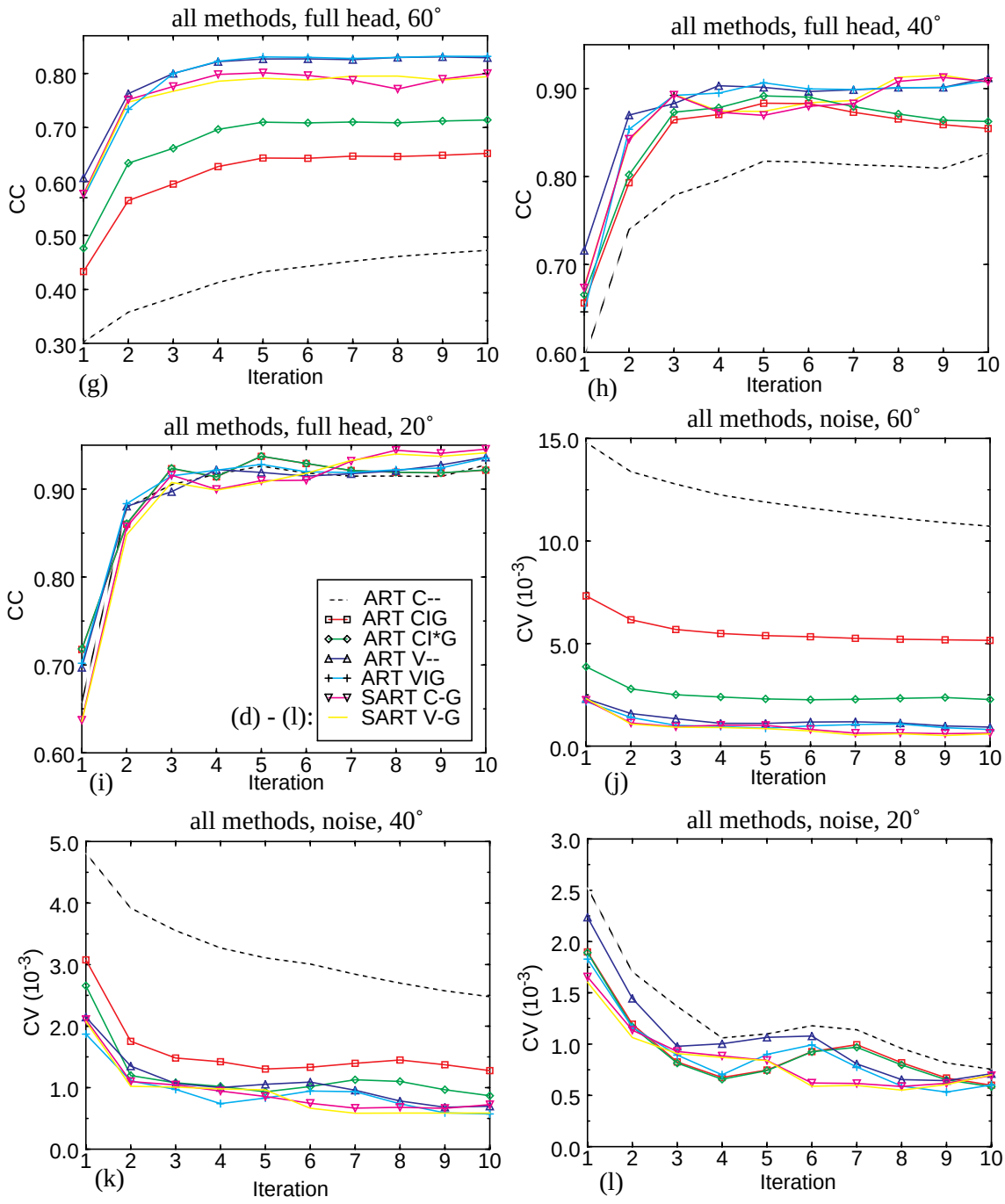
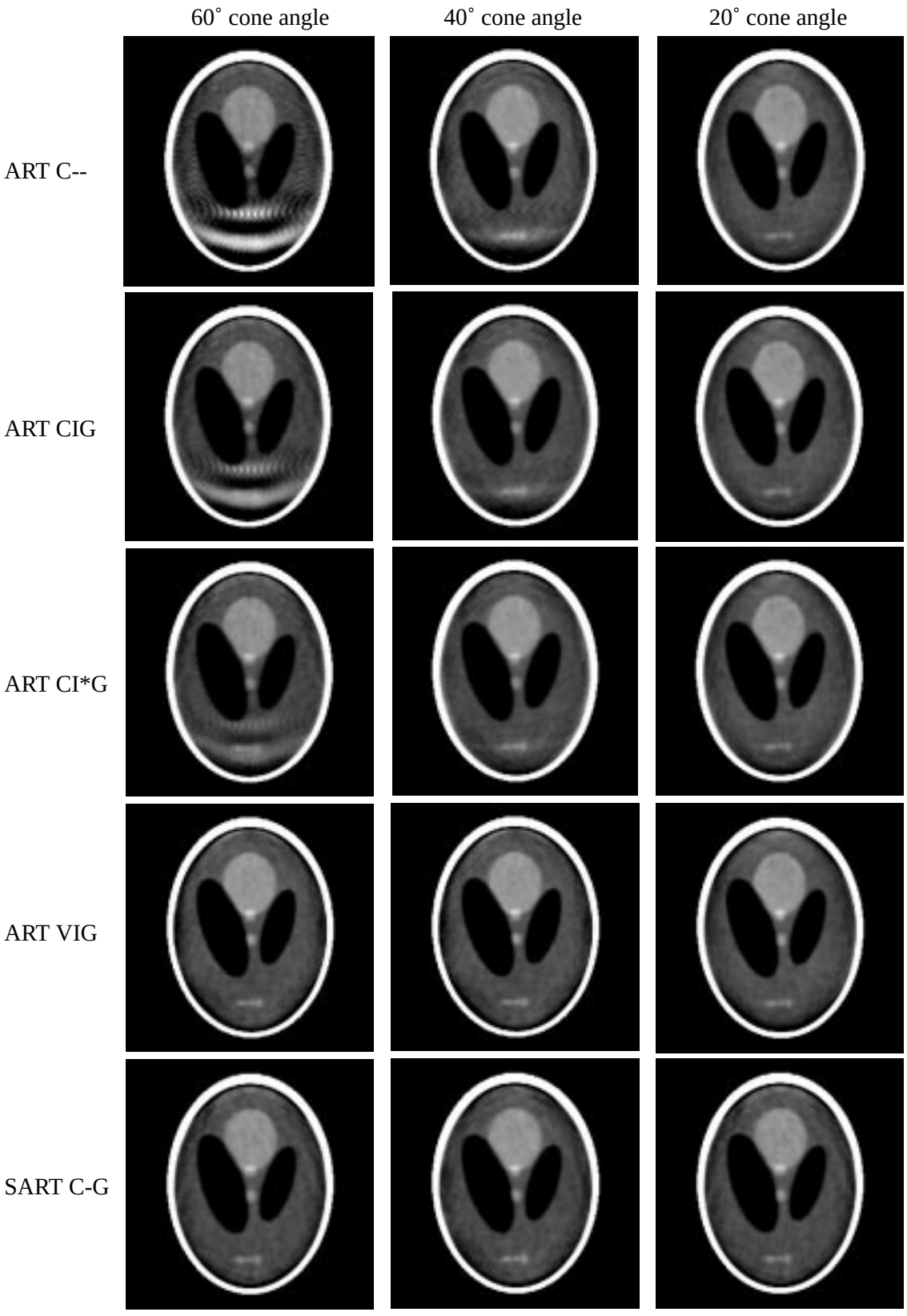


FIGURE 4.11: Continued from previous page.

FIGURE 4.12: (next page): Reconstructions of the 3D Shepp-Logan brain phantom for cone angles of 20˚, 40˚, and 60˚ with different ART and SART parameter settings.


60˚ cone angle
40˚ cone angle
20˚ cone angle
ART C--
ART CIG
ART CI*G
ART VIG
SART C-G

eliminate these settings from further plots. Finally, in Figure 4.11c, we see that SART C-G is either similar or always better than SART C--, and the same is true for SART V-G and SART V--. Thus we will be using only SART C-G and SART V-G. Also, preceding experiments revealed that SART is not dependent on the initial state of the volume. This is largely due to the circumstance that SART corrects the volume on an image basis, which provides a good initialization after the first image was spread onto the volume.

The following plots (Figure 4.11d-l) illustrate the effects of the settings of the remaining parameters on CC and CV. (Please use the legends inserted into Figure 4.11f and i). In Figure 4.11d, g, and j, we observe that when using the constant sized interpolation kernel for ART, CC and CV for reconstructions at a 60˚ cone angle improve significantly as volume initialization is made more accurate. This can also be seen in the reconstructed images shown in the first column of Figure 4.12. If the volume is initialized to zero, the small SLP tumors (and other portions of the image) are completely obliterated by aliasing artifacts (ART C--). The artifacts are reduced somewhat if the volume is initialized to the average value of one of the projection images (ART CIG). The artifacts are reduced further, but still well noticeable, if the volume is initialized to the average value of the SLP brain matter (ART CI*G). It is apparent that volume initialization alone cannot remove all artifacts. However, the reconstructions obtained with the variable sized interpolation kernel in conjunction with ART and the ones obtained with SART are all artifact-free. The plots support these observation with only SART and the ART V-methods having good CC and low CV. The plots also indicate that reconstruction quality increases when SART is used with a V-kernel instead of a C-kernel, however, the improvements are not large. At the same token, reconstruction quality also improves when the ART-V methods are used in conjunction with volume initialization and gradually increasing relaxation coefficient, but the rate of improvement is at a much smaller scale than in the ART-C case.

From the images in the second column in Figure 4.12 we observe that for a cone angle of 40˚ considerable artifacts around the tumors still remain for ART C-- and ART CIG. Again notice the improvements with more accurate volume initialization. For ART CI*G, the artifacts are more attenuated but are still visible (however, note again that ART CI*G may not always be realizable). On the other hand, with ART VIG and SART C-G, the artifacts are completely eliminated. The plots of Figure 4.11e, h, and k support these observations: the CCs are consistently higher, especially for small object detail like the SLP tumors, and the CVs are consistently lower with the ART V-methods and SART than for the ART C-methods (with ART CI*G being closest to ART V and SART).

The plots of Figure 4.11f, i, and l indicate that for a smaller cone angle of 20˚ the differences between the methods are not as pronounced as for the larger cone angles, as long as one initializes the volume at least with the average projection image value. The circumstance that ART VIG and SART maintain a marginally better CC for the SLP tumors in a quantitative sense could be relevant for automated volume analysis and feature detection. However, in a visual inspection the differences are hardly noticeable, as indicated in the reconstruction images in the third column of Figure 4.12.

4.4 Error Analysis

In Section 4.1 we used the approximate ray grid sampling rate ω_r to simplify our analysis. We shall now formally investigate the magnitude of the error committed by this approximation.

Consider Figure 4.13 where we show one of several ways to estimate the accurate ray grid sampling rate $\hat{\omega}_r$ at a volume grid position with $z=z_s$ and $y=iT_r$. Here, $i \in \aleph$ is the index of the projection pixel and T_r is the distance between the points of intersection of the lines that go from the source to the pixel boundaries with the volume slice plane being located at $z=z_s$. The method uses the perpendicular distance $\hat{T}_r$ between these two pixel bounding lines to compute $\hat{\omega}_r = 1/\hat{T}_r$ for the ray of pixel i. See the description in the caption of Figure 4.13 for a more detailed explanation.

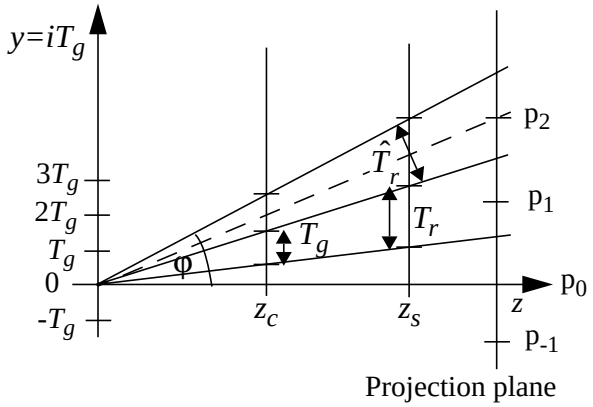


FIGURE 4.13: Computing the perpendicular ray distance $\hat{T}_r$ to estimate the accurate ray grid sampling rate $\hat{\omega}_r$. The dashed line is the ray that traverses from the source through the center of projection pixel i=2. The lines (shown in solid) that connect the pixel boundaries with the source intersect the volume slice at $z=z_s$. The distance between these intersections is T_r, and $1/T_r$ was used in Section 4.1 as the approximate ray grid sampling rate. T_g is the period of the volume grid and z_c is the slice in which $T_r=T_g$. Finally, $\hat{T}_r$ is given by the perpendicular distance of the two pixel boundary rays at $z=z_s$.

The perpendicular distance $\hat{T}_r$ is given by:

$$\hat{T}_r \approx T_r \cos\varphi \equiv \frac{z_s}{z_c} T_g \cos\varphi = \frac{z_s}{z_c} T_g \cdot \frac{z_c}{\sqrt{\langle iT_g \rangle^2 + z_c^2}} = \frac{z_s T_g}{\sqrt{\langle iT_g \rangle^2 + z_c^2}} \qquad (4.20)$$

A plot of the rayfront for which $\hat{T}_r = T_g = 1$ is shown in Figure 4.14.

The relative error $e_{stretch}$ when using T_r instead of $\hat{T}_r$ for stretching the kernel functions is

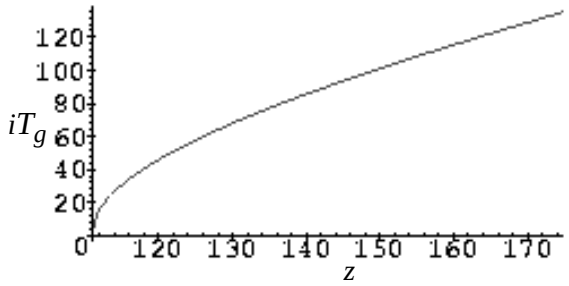


FIGURE 4.14: Shape of the rayfront with constant $\hat{\omega}_r$. Here the case of $\hat{\omega}_r = 1.0$ is shown. In the regions on the left side of the curve the ray grid sampling rate is higher than the sampling rate of the volume grid, while in the regions on the right of the curve the ray grid sampling rate is lower than the volume grid sampling rate.

given by:

$$e_{stretch} = T_r / \hat{T}_r = \langle \frac{z_s}{z_c} T_g \rangle / \langle \frac{z_s T_g}{\sqrt{\langle iT_g \rangle^2 + z_c^2}} \rangle = \frac{\sqrt{\langle iT_g \rangle^2 + z_c^2}}{z_c} \tag{4.21}$$

This error is plotted in Figure 4.15. We observe that further away from the cone center the kernel is stretched too much if the curved ray front is approximated by a planar rayfront to estimate ω_r. This means that the grid signal is overly smoothed when sampled. The maximum error of 15% occurs at the cone boundary, i.e., here the kernel is stretched 15% more than necessary.

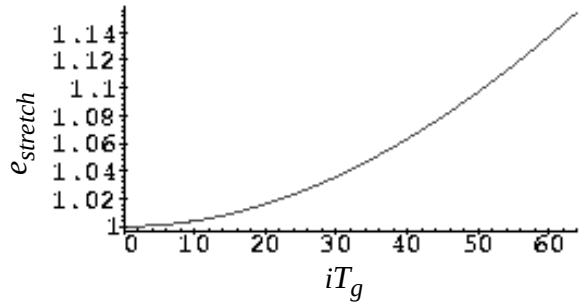


FIGURE 4.15: The relative error $e_{stretch}$ when using T_r instead of $\hat{T}_r$ for stretching the kernel functions.

CHAPTER 5

CONE-BEAM RECONSTRUCTION WITH ART: SPEED ISSUES

The previous chapter has shown how we can make algebraic methods accurate for the general cone-beam case. This chapter will demonstrate how they can be made efficient. The ultimate goal is to reduce the cost of algebraic methods to a level at which they become feasible for routine clinical applications. A good measure to determine this level of feasibility is the cost of the popular Filtered Backprojection (FBP) algorithm. However, we don't necessarily need to match or exceed the computational expense of FBP (although that would, of course, be the utmost desirable). We just need to get reasonably close to the cost of FBP, as the many advantages that algebraic methods have over FBP will offset a small amount of extra computational work (at least in those applications where ART actually produces better results).

Most of the computational expense of ART is spent for grid projection and backprojection, thus our main effort should go into optimizing these operations. It turns out that the computational cost and accuracy of grid projection is greatly affected by the perspective cone-beam geometry. In the following sections, we will give a detailed description of two new, highly accurate projection algorithms, one voxel-driven and one ray-driven, and analyze their efficiency in both the parallel-beam and cone-beam setting. Although other voxel-driven projectors [67] and ray-driven projectors [31][33] have been described, these algorithms are only efficient for the parallel-beam case or do not allow the stretched interpolation kernels prescribed in Section 4.1 as necessary for accurate cone-beam reconstruction. Furthermore, our new voxel-driven perspective projection algorithm is considerably more accurate than the one described by Westover [67]. Our ray-driven algorithm, on the other hand, is a 3D extension of the 2D algorithm proposed by Hanson and Wecksung [22].

However, a fast projection algorithm is not enough. We must also reduce the actual complexity of the overall projection-backprojection framework (refer to equation (1.6) and our comments above). Ideally, we only want to do the computational equivalent of one projection operation per image instead of one projection and one backprojection. This can only

be achieved by re-using some of the earlier computed results for later calculations, which is a technique termed caching. Our paper will give caching schemes for both ART and SART, which will bring the computational cost of these two popular algebraic methods close to the theoretical cost of FBP methods. These reduction factors will then be discussed in the framework of equation (1.6), and a final feasibility analysis will contrast algebraic methods with FBP in terms of cost to determine ART's utility in a clinical setting.

Thus the outline of this chapter is as follows. After presenting relevant previous work in Section 5.1, we describe, in Section 5.2, a new voxel-driven projection algorithm for cone-beam that is considerably more accurate for perspective projection than existing ones, but does not improve the state of the art in terms of speed. With this in mind, we present, in Section 5.3, a new ray-driven projection algorithm for cone-beam ART and SART that is both accurate and efficient. Next, in Section 5.4, we discuss various caching schemes that reduce the overall complexity of ART and SART and speed these methods up considerably. Finally, Section 5.5 puts everything together and presents a variety of results obtained with our ART testbed software. We conclude, in Section 5.6, with an analysis of the feasibility of algebraic methods for routine clinical use. Two appendices have been added, in Section 5.7, to discuss two side-issues raised in the main text.

5.1 Previous Work

While the concept of representing a volume by a field of interpolation kernels and pre-integrating a 3D kernel into a 2D footprint is common to all existing splatting-type implementations, the strategy chosen to map the footprint table onto the screen (in the voxel-driven approach) or to map the rays into the footprint table (in the ray-driven approach) varies. The mapping task is facilitated since we only use spherically symmetric kernels and cubic grids, which yields a circular footprint. For voxel-driven splatting, both Westover [67] and Matej and Lewitt [33] simply map the circular footprint to the projection screen for one voxel and use incremental shifts for the remaining voxels at that projection angle. This, however, is only correct for parallel projections, since in perspective projection the elliptical shape and size of the footprint is different for every voxel. (More detail is given in Section 5.2.)

In the case of ray-driven splatting we again assume a spherically symmetric interpolation kernel, but here the approaches are more diverse. For instance, Lewitt [31] computes the magnitude of the cross-product of the ray unit vector with the vector from a point on the ray to the voxel center. This yields the perpendicular distance of the ray to the voxel center which can be used to index a 1D footprint table storing the radially symmetric projection of the 3D kernel. Efficient incremental algorithm can then be used to find all other voxel distances along the ray. This approach, however, is not appropriate for cone-beam reconstruction, as it does not allow independent footprint stretching in the two ray sheet directions. In another approach, Matej and Lewitt [33] decompose the voxel grid into a set of 2D slices. Here, the orientation of the slices is that orientation most parallel to the image plane. Recall that a footprint is the pre-integrated kernel function in the direction of a ray, thus a footprint is not necessarily planar with the slice planes. The authors project this footprint function onto a slice plane which results in an elliptical footprint. Since in parallel projec-

tion all rays for a given projection angle have the same angle with the volume slices, this re-mapped elliptical footprint can be used for all slices and for all rays that are spawned for a given projection orientation. Simple incremental algorithms can be designed to trace a ray across the volume slices, computing all indices into the elliptical footprints that are intersected. However, for perspective projection, every ray has a different orientation, necessitating a footprint re-mapping for every ray, which is inefficient both to compute on the fly and to store. A more appropriate approach was outlined for the 2D case by Hanson and Wecksung [22]. These authors model a 2D ray as an implicit line equation. If one runs a line parallel to the ray across the center of a given voxel, then the offset difference of the equations of these two lines yield the perpendicular distance of the ray to the voxel center. This distance can then be used to index a 1D footprint table. Our ray-driven approach is a 3D extension of this algorithm, optimized for speed, that enables the efficient use of the same footprint table for all projection rays everywhere in the volume.

5.2 An Accurate Voxel-Driven Splatting Algorithm for Cone-Beam ART

Let us first introduce some terminology. As suggested by Crawfis and Max [8], we can think of the interpolation kernel footprint as a polygon with a superimposed texture map that is placed in object (volume) space. Here, the texture map is given by the projected kernel function, i.e. the array of line integrals. For the remainder of our discussion we will refer to the footprint in object space as the *footprint polygon*, while the projection of the footprint polygon onto the image plane will be called the *footprint image*. Recall that splatting accumulates the same value in a pixel on the image plane as a ray would accumulate when traversing the volume. Thus, when projecting the footprint polygon to obtain the line integral for the pixel in the footprint image we must ensure that we position the footprint polygon orthogonal to the direction of the sight-ray in object space. The line integrals are retrieved from the footprint table by indexing it at the ray-footprint polygon intersection point. Thus, for splatting to be accurate, the 2D footprint must be mapped to the pixel as if the ray emanating from the pixel had traversed it at a perpendicular angle. Only then does the looked-up pre-integrated integral match the true kernel integration of the ray. Westover's perspective extension to voxel-driven splatting violates this condition at three instances:

- He does not align the footprint polygon perpendicularly to the voxel center ray when calculating the projected screen extent. Instead he aligns it parallel to the screen and stretches it according to the perspective viewing transform.
- When mapping the footprint to the screen pixels he uses a linear transform instead of a perspective one.
- The footprint polygon is not rotated for every mapped pixel such that the corresponding pixel ray traverses it at a perpendicular angle.

While the error for the last approximation is rather small (see Section 5.7.1), the former two are more significant. The first approximation computes footprint screen extents that are smaller than the actual ones. For example, for a cone angle of 30° and a 128^3 volume the

maximum error ratio between correct and approximate footprint extent is 1.15 and the maximum absolute difference between the two is 1.46 pixels (see Section 5.7.2). Here, the absolute error is largest for those voxels that are located close to the source *and* close to the view cone boundary. It causes the footprints of these voxels to cover less area on the projection plane than they really should. The second approximation has a similar effect. In that case, the mapping of the footprint table entries to the screen is slightly squashed. Again, voxels close to the source and close to the view cone boundary are most affected.

Consider now Figure 5.1, where we illustrate a new and accurate solution for perspective voxel-driven splatting. For simplicity of drawing, we show the 2D case only. Note that the coordinate system is fixed at the eye point. To splat a voxel $v_{x,y,z}$, it is first rotated about the volume center such that the volume is aligned with the projection plane. Then the footprint polygon is placed orthogonal to the vector starting at the eye and going through the center of $v_{x,y,z}$. Note that this yields an accurate line integral only for the center ray, all other rays traverse the voxel kernel function at a slightly different orientation than given by the placement of the 2D (1D in Figure 5.1) footprint polygon in object space. Thus the first error in Westover's approximation still survives. This error, however, can be shown to be less than 0.01, even for voxels close to the source.

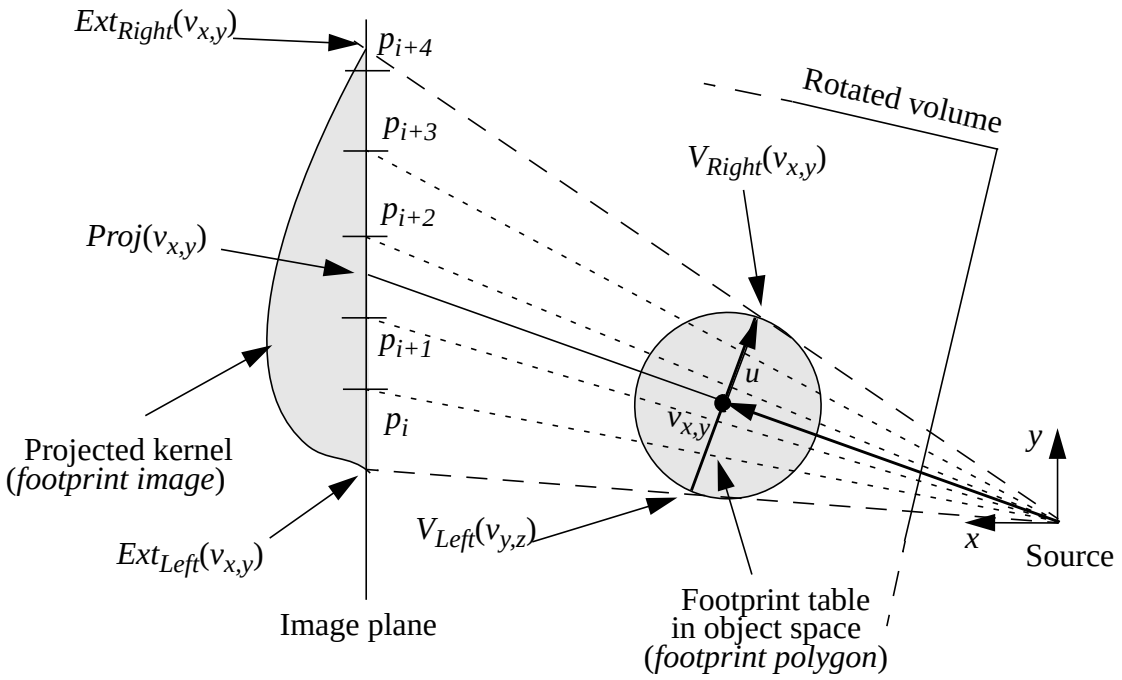


FIGURE 5.1: Perspective voxel-driven splatting: First, the footprint polygon of voxel $v_{x,y}$ is mapped onto the image plane, then the affected image pixels $p_i...p_{i+4}$ are mapped back onto the footprint table.

The coefficients of the footprint polygon's plane equation are given by the normalized cen-

ter ray (the vector *source*-$v_{x,y,z}$). From this equation we compute two orthogonal vectors *u* and *w* on the plane (only *u* is shown in Figure 5.1). Here, *u* and *w* are chosen such that they project onto the two major axes of the image. Using *u* and *w*, we can compute the spatial *x,y,z* positions of the four footprint polygon vertices in object space ($V_{Right}(v_{x,y})$ and $V_{Left}(v_{x,y})$ in the 2D case depicted in Figure 5.1). These four vertices are perspectively projected onto the image plane. This yields the rectangular extent of the footprint image, aligned with the image axes ($Ext_{Right}(v_{x,y})$ and $Ext_{Left}(v_{x,y})$ in the 2D case). By expressing the intersections of the pixel rays with the footprint polygon in a parametric fashion, we can then set up an incremental scheme to relate the image pixels within the footprint image with the texture map entries of the footprint table.

The computational effort to map a footprint polygon onto the screen and to set up the incremental mapping of the pixels into the footprint table is quite large: Almost 100 multiplications, additions, and divisions, and two square root operations are necessary. No incremental scheme can be used to accelerate the mapping of neighboring grid voxels. The high cost is amplified by the fact that the expensive mapping has to be done at $O(N)=O(n^3)$. And indeed, in our implementation, perspective projection was over twice as expensive than parallel projection.

5.3 A Fast and Accurate Ray-Driven Splatting Algorithm for Cone-Beam ART

We saw in the previous section that perspective voxel-driven splatting can be made accurate, however, the expense of perspective voxel-driven splatting seems prohibitive for use in cone-beam reconstruction. In this section we take advantage of the fact that, in contrast to voxel-driven approaches, ray-driven methods are generally not sensitive to the non-linearity of the perspective viewing transform. It can thus be expected that ray-driven splatting is more advantageous to use in the perspective cone-beam situation. The new ray-driven approach is in some respect a 3D extension to the 2D algorithm sketched by Hanson and Wecksung [22] and will work both for constant-size kernels as used in cone-beam SART and variable-size kernels as required for cone-beam ART.

5.3.1 Ray-Driven Splatting with Constant-Size Interpolation Kernels

In ray-driven splatting, voxel contributions no longer accumulate on the image plane for all pixels simultaneously. In contrast, each pixel accumulates its raysums separately, which makes it also more suitable for ART than voxel-driven splatting. Our algorithm proceeds as follows. The volume is divided into 2D slices formed by the planes most parallel to the image plane. When a pixel ray is shot into the 3D field of interpolation kernels, it stops at each slice and determines the range of voxel kernels within the slice that are traversed by the ray. This is shown in Figure 5.2a for the 2D case: The ray originating at pixel p_i pierces the volume slice located at x_s at $y=y(i,x_s)$. The voxel kernels within the slice x_s that are intersected by the ray are given by the interval $[\text{Ceil}(y_{Left}(i,x_s)), \text{Floor}(y_{Right}(i,x_s))]$. We compute $y_{Right}(i,x_s)$ as:

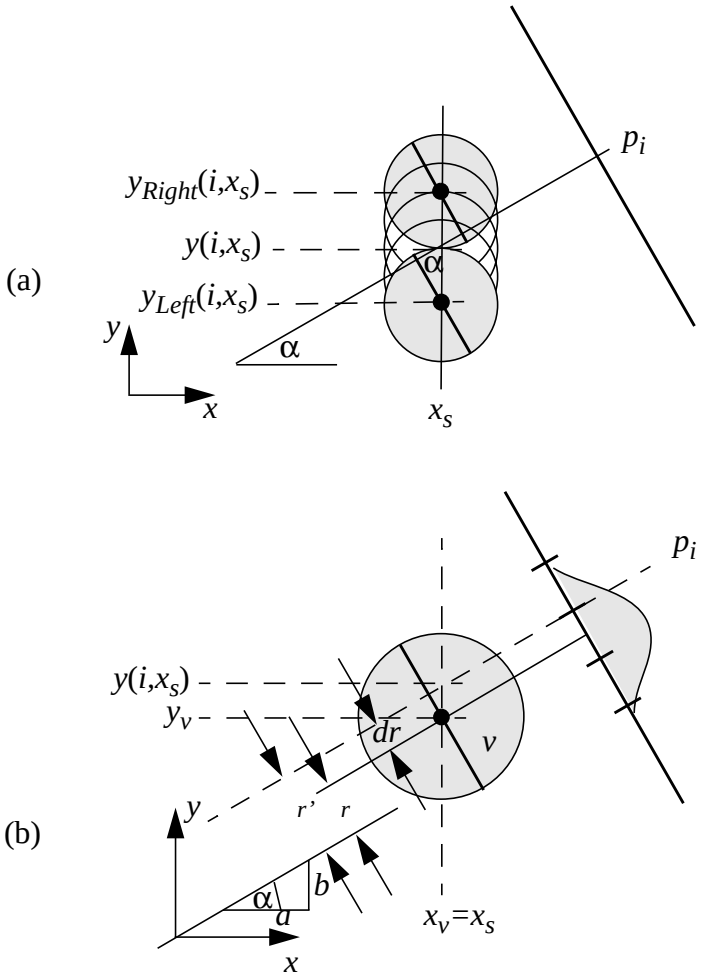


FIGURE 5.2: Ray-driven splatting: (a) Determining the range of voxels within a given volume slice plane that are traversed by a ray originating at pixel p_i. (b) Computing the index *dr* into the footprint table.

$$y_{Right}(i, x_s) = y(i, x_s) + \frac{extent_{kernel}}{\cos(\alpha)} \tag{5.1}$$

where α is the inclination of the ray. The computation for $y_{Left}(i,x_s)$ is analogous. After determining the active voxel interval $[y_{Left}(i,x_s), y_{Right}(i,x_s)]$ we must compute the indices into the voxel footprint table. This can be efficiently implemented by realizing that the index into the footprint table of a grid voxel *v* located at coordinates (y_v,x_v) is given by the distance *dr* of the two parallel lines (planes in 3D) that traverse *v*'s center point and the slice

intersection point of the ray at $y(i,x_s)$, respectively (see Figure 5.2b). One finds:

$$dr = a \cdot x_s + b \cdot y(i, x_s) - a \cdot x_s - b \cdot y_v = (b \cdot (y(i, x_s) - y_v)) \tag{5.2}$$

where a and b are the coefficients of the implicit line equation $a \cdot x_s + b \cdot y(i, x_s) = 0$ of the ray and are also given by the components of the (normalized) ray vector. Maintaining the variables $y_{Left}(i,x)$, $y_{Right}(i,x)$, and dr along a ray can all be done using incremental additions.

For the 3D case, we need to replace the linear ray by two planes. A 3D ray is defined by the intersection of two orthogonal planes cutting through the voxel field. The normal for one plane is computed as the cross product of the ray and one of the image plane axis vectors. The normal of the second plane is computed as the cross product of the ray and the normal of the first plane. Thus, the two planes are orthogonal to each other and are also orthogonal to the voxel footprint polygons. Thus the ray pierces the footprint polygon in a perpendicular fashion, as required. Intersecting the horizontal plane with a footprint polygon and using plane equations in the spirit of equation (5.2) results in the horizontal row index dr_{row} into the footprint table, while the intersection with the vertical plane yields the vertical column index dr_{col}. Using these two indices, the value of the ray integral can be retrieved from the 2D footprint table. Note that the two orthogonal directions of the indices, dr_{col} and dr_{row}, on the footprint polygon plane allow us to implement the bi-directional footprint stretching required for the variable-size interpolation kernels in cone-beam ART.

There are now three nested loops: The most outer loop sets up a new ray to pierce the volume, the next inner loop advances the ray across the volume slice by slice and determines the set of voxels traversed per slice, and finally, the most inner loop retrieves the voxel contributions from the footprint tables. For perspective projection, the plane equations have to computed for every ray. This amounts to about 50 extra additions, multiplications, and divisions, and three square roots per pixel. The cost of advancing a ray across the volume and determining the footprint entries is comparable to the cost of rotating a kernel and splatting it onto the image plane in the orthographic voxel-driven approach. The ray-driven approach changes the splatting algorithm from voxel order to pixel order. Thus, the most outer loop is of $O(n^2)$. This has the advantage that the complexity of any extra work that has to be done for perspective projection (e.g. recomputing the two planes that define the ray in 3D) is roughly one order of magnitude less than in voxel-driven splatting. Note also that ray-driven splatting does not introduce inaccuracies. As a matter of fact, it prevents the indexing errors in the voxel-driven approach by design.

5.3.2 Ray-Driven Splatting with Variable-Size Interpolation Kernels

We have discussed in the previous chapter that the aliasing artifacts caused by the diverging set of rays in cone-beam can be eliminated by progressively increasing the interpolation filter width (or magnitude, in backprojection for $s(z)<z_c$) as a linear function of ray depth. This requires us to express the sampling rate ω_r of the arrangement of rays in grid coordinates (x,y,z). Once the function ω_r is known, we can determine the required interpolation filter width or magnitude at each location along a ray. We saw in Figure 4.4 that ω_r is con-

stant along the curved rayfront iso-contours and decreases linearly with increasing distance of the iso-contours from the source. This linear dependence on ray depth means that each voxel kernel must undergo a non-uniform distortion along a ray. However, since we use identical, pre-integrated kernels in the form of 2D footprint polygons, we cannot realize this non-uniform distortion function. Hence, as an approximation, we only estimate ω_r at the location of each kernel center and distort the generic 2D footprint.

Consider now Figure 5.3. The coordinates of an image pixel can be expressed as $p_{ij}=image\_origin+iu+jv$, where u, v are the orthogonal image plane axis vectors. The grid of diverging rays is organized into horizontal and vertical sheets, or cutting planes, that intersect the image plane and are spaced by u and v. The ray grid sampling rate ω_r is then a 2D vector (ω_{ru}, ω_{rv}) that is related to the local sheet spacings. Figure 5.3 illustrates how ω_{rv} is calculated. Here, we see the two horizontal cutting planes cp_j and cp_{j+1} embedding the rays $r_{i,j}$ and $r_{i,j+1}$, respectively. To approximate $T_v=1/\omega_{rv}$ at the location $(x_v\, y_v\, z_v)$ of the kernel center of voxel $v_{x,y,z}$, we measure the distance between cp_j and cp_{j+1} along the vector orthogonal to cp_j passing through $(x_v\, y_v\, z_v)$. This distance can be written as $T_v=ax+by+cz+k$ where $(a\ b\ c)$ is a linear function of (x,y,z) and the plane equations of cp_j and cp_{j+1}, and can thus be updated incrementally for all intersected voxels along a ray. If we select the horizontal and vertical cutting planes such that the image plane vectors u and v, respectively, are embedded in them, then we can simply stretch the footprint polygon by a scale factor of magnitude T_v to achieve the proper lowpassing in that ray grid direction. (Recall that, in forward projection, we also have to scale the kernel's amplitude by a factor $1/T_v$.) An analogous argument can be given for the vertical cutting planes and $T_u=1/\omega_{ru}$. Also recall that we only stretch the footprint polygon if $T_u>1$ or $T_v>1$. If, in forward projection, $T_u\leq 1$ or $T_v\leq 1$ then the ray grid sampling rate in that direction is sufficient such that no aliasing can occur. However, in backprojection we do need to scale the magnitude of the kernel whenever $T_u<1$ or $T_v<1$. Here, the scale factor is $1/T_v$ or $1/T_u$, respectively.

Note that in order to preserve orthogonality of the two cutting planes, in the general case one cannot achieve that u lies in the horizontal cutting planes and, at the same time, v lies in the vertical planes. But since we flip the main viewing direction as soon as the angle between the ray direction and the major viewing axis exceeds 45˚, the angular deviation of the true orientation of the cutting plane and the correct orientation is small (less than 5 degrees).

The method outlined above can be efficiently implemented by using an incremental scheme that requires about one addition per voxel in each slice for computing the distance between two cutting planes. However, due to the fact that u lies in the x-z plane and v is aligned with the y-axis of the volume grid, we may employ a simpler method that does not compute the *perpendicular* distance between two adjacent cutting planes, but uses the *slice-projected* distance between two adjacent cutting planes as they intersect with the volume slice most parallel to the projection plane. This approximation is shown for the 2D case in Figure 5.4, (we have also looked at this approximation, in a slightly different context, in Section 4.4). In this figure, T_{corr} is the distance measured by the scheme described first, while T_{approx} is the volume slice-projected measurement. The error is given by $T_{corr} \approx T_{Tapprox} \cdot \cos\varphi_i$. This means that the simpler method underestimates the grid sampling rate by some amount.

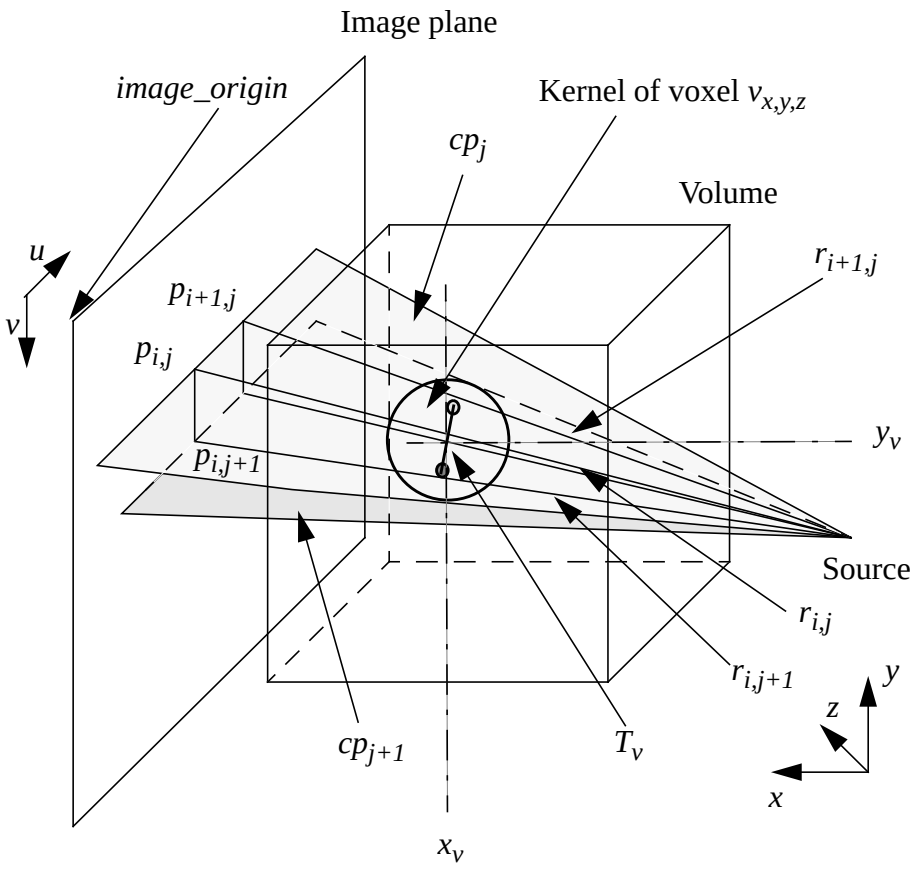


FIGURE 5.3: Determining the local sampling rate of the arrangement of diverging rays. The arrangement of rays traverses the volume grid in two orthogonal sets of ray sheets (two horizontal sheets, i.e. the cutting planes cp_j and cp_{j+1}, are shown). Each 3D ray is part of one sheet in each set and is defined by the intersection of these two sheets. Depending on the location of the kernel with respect to each sheet set, the 2D kernel footprint must undergo different distortions in its two principal coordinate axes.

In the case of ω_{rv}, the maximum error occurs for voxels on the view cone boundary. Here, for a cone half-angle φ_c=30˚, the simpler method would choose a kernel that is about 1/0.86=1.15 times larger than it needs to be, and thus lead to a greater amount of lowpassing of voxel $v_{x,y,z}$ in the *v*-direction than is required. However, the fact that the factor cos(φ) is rather small and approaches values close to 1.0 quickly as we move away from the view cone boundary, makes this approximation a justifiable one. In the case of ω_{ru}, which determines the kernel stretching factor in the *x*-*z* plane, the error can get a bit larger. Here, the rays can meet the volume slice plane most parallel to the viewing plane at an angle of up to 45˚. Greater angles are avoided since we flip the major viewing direction as soon as an angle of 45˚ is exceeded, as was mentioned above. Thus the error T_{approx}/T_{corr} when determining the kernel stretch factor for the *u*-direction can grow up to 1/cos(45˚)=1.41.

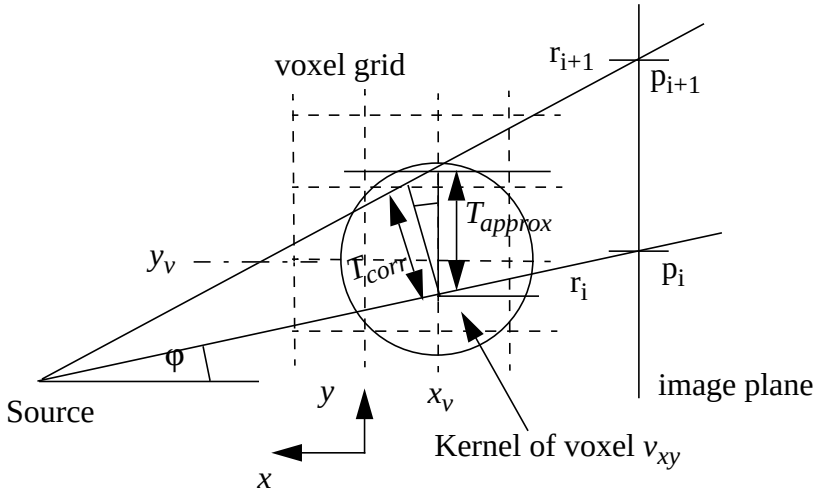


FIGURE 5.4: Error for different schemes of measuring the ray grid sampling rate.

5.4 Caching Schemes for Faster Execution of ART and SART

In the previous section, we have discussed a ray-driven projection algorithm that minimizes the number of *necessary* calculations per projection by utilizing fast incremental ray traversal schemes. In this section we will investigate to what extent previously computed results can be re-used, i.e. cached, so the number of *redundant* calculations can be minimized.

Caching can be performed at various scales, with the largest scale being iteration-based, in which all weights are pre-computed and stored on disk. The number of weights to be stored can be estimated as follows. If we only consider voxels in the spherical reconstruction region then the total number of relevant voxels $N \approx 0.5n^3$. With a square footprint extent of 4.0, the average number of rays traversing a footprint polygon is 16. Thus the number of relevant weights per projection is $8n^3$. For a number of projections S=80 and a voxel grid of 128^3 voxels, the total number of relevant weight factors is then about 1.3G floating point values, 5.3GB of actual data. This is clearly too much to hold in RAM. On the other hand, if we just held the coefficients for one projection at a time, we would need 67MB of RAM. This is in addition to the volume data and other data structures, but is feasible with today's workstations. However, then we would have to load a 67M file for every new projection that we work on. It is likely that the disk transfer time exceeds the savings in that case. In addition, the memory demands grow dramatically for larger volumes, since the number of weights to store is 8 times the number of voxels.

Since caching on both the iteration and the image level is not practical, one may exploit

caching on the ray level. ART is an easy candidate for this form of caching since a pixel projection is immediately followed by a pixel backprojection. So one can just cache the weight factors calculated in the projection step for the backprojection step, and speedups of close to 2.0 can be realistically expected with only little memory overhead.

For SART, two special problems need to be addressed, one has to do with the use of a ray-driven projection algorithm, the other deals with caching. While ART was easy to pair with a ray-driven projection algorithm since it itself is ray-driven, the backprojection step of SART is inherently voxel-based and requires some adaption in order to limit memory requirements. In a brute-force implementation, a backprojection would require two additional volumes, one to store the weight accumulation and one to store the correction accumulation per voxel (see equation (2.17)). Only after all backprojection rays have been traced, the correction buffer of each voxel can be divided by the weight buffer to form the voxel correction value. Thus we need extra memory to hold $2n^3$ floating point values. We can reduce this amount by an order of magnitude to $2n^2$ by tracing all rays simultaneously in form of a ray-front. Since the projection algorithm is slice-based, i.e. it considers all voxels in one volume slice before it moves to the next, we can step the entire ray-front from one slice to the next, buffering and updating only the voxels within the active volume slice.

In SART, the caching of weights computed during projection is also more difficult, since first an entire image must be projected before the grid corrections can be backprojected. Thus, at first glance, we may only be able to use caching at an image level. This would require us to allocate memory space for $8n^3$ floating point weights, e.g. $32n^3$ bytes, which is in addition to other memory requirements. While for n=128, this may be feasible for an average workstation (the required memory is then 67MB), for n=256 the memory requirements would be a hefty 536MB, which may not be readily available in most workstations. Thus, in real world applications, caching on the image-level is not feasible, at least with today's workstations, and one must design a caching scheme at a finer granularity.

For this purpose, we designed a scheme that keeps two active slabs, composed of sheets of voxel cross-sections. These voxel cross-sections are formed by intersecting the voxel kernels by consecutive horizontal cutting planes (recall Figure 5.3). In this scheme, one active slab, $slab_p$, is composed of voxels that are currently projected, while the other, $slab_b$, is composed of currently backprojected voxels. The correction and weight buffers are kept with $slab_b$, and $slab_b$ is always trailing $slab_p$. At first, $slab_p$ caches the weights computed in the projection step. Then, as $slab_p$ moves upward in the volume, voxels on the bottom of $slab_p$ have eventually been completely projected and can be removed from $slab_p$ and added to $slab_b$, along with all cached weights. A linked list can be used to facilitate passing the data. As $slab_b$ moves upward as well, voxels at the bottom of $slab_b$ can eventually be updated by the accumulated correction buffer term and be removed from $slab_b$. The width of a slab is about four sheets. Recalling that a voxel is traversed by about four rays in each sheet, the memory complexity for the slab buffers is roughly $4\,(4 + 12)\,n^2 = 64n^2$. This includes the memory for the correction and accumulation buffers of $slab_b$. Thus we would require approximately 4M of memory for a 128^3 volume. Note that this scheme goes well with the variable-size voxel kernels since here the slab width is constant for $z>z_c$.

5.5 Results

Table 5.1 lists the runtimes of the various ART and SART incarnations that were discussed in the previous sections. The runtimes were obtained on an SGI Indigo^2 workstation and refer to a reconstruction based on 80 projections with a cone angle γ=40˚.

Method	beam	splatting method	cache	var-size kernel	T_{corr} (sec)	T_{iter} (hrs)	T_{3iter} (hrs)
SART	parallel	voxel	-	-	35.3	0.78	2.35
SART	parallel	ray	-	-	47.1	1.04	3.14
SART	cone	voxel	-	-	144.9	3.22	9.66
SART	cone	ray	-	-	60.9	1.35	4.05
SART	cone	ray	-	✓	73.2	1.63	4.89
SART	cone	ray	✓	-	43.6	0.97	2.90
SART	cone	ray	✓	✓	52.4	1.16	3.49
ART	cone	ray	-	-	54.2	1.20	3.60
ART	cone	ray	-	✓	68.2	1.51	4.53
ART	cone	ray	✓	-	30.3	0.67	2.01
ART	cone	ray	✓	✓	38.1	0.85	2.55

TABLE 5.1: Run-times for SART, using both voxel-driven and ray-driven splatting, and for ART using ray-driven splatting (voxel-driven splatting is not applicable for ART). The effect of caching and variable-size interpolation kernels on the run-time is also shown. (T_{corr}: time for one grid correction step, consisting of one projection and one backprojection, T_{iter}: time for 1 iteration (assuming 80 projection images and a cone angle of 40˚), T_{3iter}: time for 3 iterations, the minimum time to obtain a reconstruction of good quality. Timings were obtained on a SGI Iris Indigo^2 with a 200MHz RS4000 CPU.)

Let us first look at the SART correction algorithm. We observe, in Table 5.1 that for parallel-beam reconstruction with SART the voxel-driven approach is about 33% faster than the ray-driven approach. Hence, it is more advantageous in the parallel-beam case to perform the grid projection in object-order (i.e. to map the footprint polygons onto the screen) than to perform the projection in image-order (i.e. traverse the array of footprint polygons by the pixel rays). The computational savings in the voxel-driven algorithm for parallel-beam pro-

jection come from the fact that here the footprint-screen mapping is much simpler than the mapping described in Section 5.2, since the perspective distortion does not have to be incorporated. In cone-beam reconstruction, on the other hand, the situation is reversed in favor of the ray-driven projector. Here, the speedup for using the ray-driven projector over the voxel-driven projector in SART is about 2.4. Thus, since for ART the use of the image-based voxel-driven splatting algorithm is not practical anyhow, we conclude that cone-beam reconstruction should always be performed with ray-driven projectors.

Now let us investigate the computational differences of ART and SART and the effects of caching and variable-size splatting kernels on run-time. Comparing the costs for SART and ART, we notice that uncached SART is about 12% slower than uncached ART. This is due to the extra computations required for weighting the corrections before a voxel update and the overhead for managing the additional data structures. The timings also indicate that the use of a depth dependent kernel size incurs about a 25% time penalty for ART and 20% for SART. In terms of the benefits of caching, we notice that the straightforward caching for ART speeds reconstruction by a factor of 1.78, while the more involved caching for SART achieves a speedup of 1.4. The speedups for caching in conjunction with the variable-size kernels are similar. Since the reconstruction results for SART using constant sized kernels and ART using variable-size kernels looked similar, it makes sense to compare these two methods as well. In this respect, ART with variable-size kernels and easy-to-implement caching is about twice as fast as uncached SART. However, if SART is enhanced with elaborate caching schemes, this speed advantage shrinks to a factor of 1.15

5.6 Cost and Feasibility of Algebraic Methods: Final Analysis

So how does the computational effort of cached cone-beam ART compare with the effort required for FBP-type methods? We shall conduct a rough estimate by using the following results:

- Cached ART requires the time-equivalent of 1.12 projections per projection/back-projection operation. This factor is computed as follows (using adaptive kernels). As indicated by the results listed in Table 5.1, a cone-beam projection with ART takes half the time of a projection/backprojection operation with *uncached* ART, i.e., 68.2/2s=34.1s. A cone-beam projection/backprojection with *cached* ART takes 38.1s. This is 1.12 times more than just a projection. Hence, a_{Alg}=2/1.12=0.56.
- ART requires only half the number of projections M that are needed for FBP (shown by [19]).
- ART must conduct three iterations to achieve a reconstruction of satisfactory quality (revealed in Section 4.3).
- FBP must perform a convolution step that bears the cost of some fraction a_{FBP} of a projection operation.
- The time required for a projection operation T_{proj} is similar for FBP and ART.

Let us now recall the cost equation (1.6) from Section 1.1.3:

$$\frac{Cost\,(Algebraic)}{Cost\,(FBP)} \;=\; \frac{2 \cdot a_{Alg} \cdot I \cdot M_{Alg}}{(1 + a_{FBP}) \cdot M_{FBP}}$$

Incorporating the results listed above we get:

$$\frac{Cost\,(Algebraic)}{Cost\,(FBP)} \;=\; \frac{2 \cdot 0.56 \cdot 3}{(1 + a_{FBP}) \cdot 2} \;=\; \frac{1.68}{(1 + a_{FBP})} \tag{5.3}$$

If we want the costs of the two methods to be equal, then the filtering operation step of FBP must take at least 68% of a projection operation. However, if we grant the higher reconstruction quality obtained with ART an extra 20% computation time, then we get:

$$a_{FBP} \;=\; \frac{1.68}{1.2} - 1 \;=\; 0.4 \tag{5.4}$$

This is certainly a realistic value. We may conclude from this — admittedly approximate — analysis, that we have come very close to making ART a serious contender to FBP in the clinical arena.

5.7 Error Analysis of Westover-Type Splatting

This section serves more as an appendix to this chapter. We have mentioned, in Section 5.2, that Westover's perspective extension to voxel-driven splatting commits the following errors when mapping a footprint to the projection screen:

- It does not align the footprint polygon perpendicularly to the voxel center ray when calculating the projected screen extent. Instead, it aligns the polygon parallel to the screen and stretches it according to the perspective viewing transform.
- When mapping the footprint to the screen pixels it uses a linear transform instead of a perspective one.
- The footprint polygon is not rotated for every mapped pixel such that the corresponding pixel ray traverses it at a perpendicular angle.

We now present a quantitative analysis if these errors.

5.7.1 Errors from non-perpendicular traversal of the footprint polygon

This section addresses the last error in the list given above. Figure 5.5 shows, for the 2D case, the footprint polygon being aligned perpendicularly to the center ray.

We observe that pixel rays other than the center ray traverse the footprint polygon at an

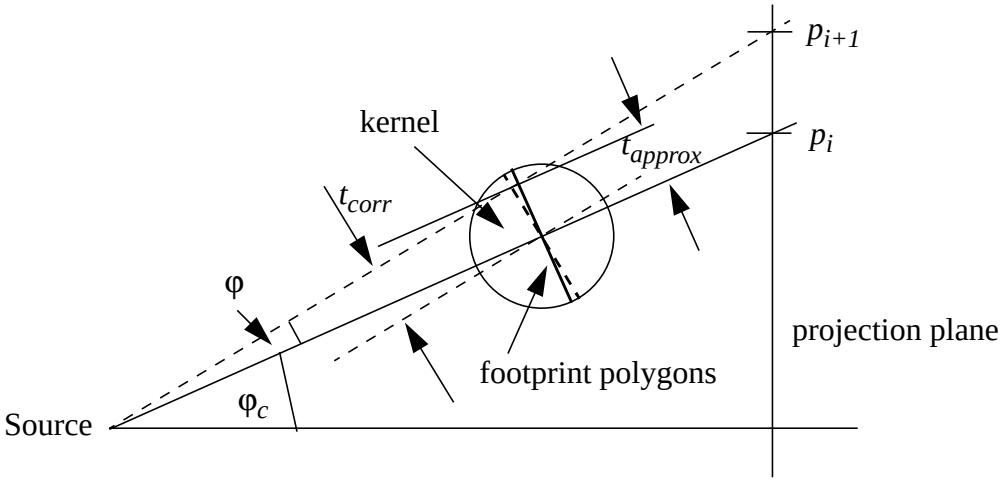


FIGURE 5.5: Errors from non-perpendicular traversal of footprint polygon

oblique angle. However, they should really have the footprint polygon aligned as depicted by the dotted line. The error when computing the lookup table index *t* for these rays is given by:

$$t_{corr} = t_{approx} \cdot \cos\varphi \tag{5.5}$$

where t_{corr} is the correct index and t_{approx} is the computed index. The angle φ is largest for voxels close to the eye and at the center of the projection plane (φ is largest there). For a cone half-angle of γ=30˚ and a 128^3 volume, the angle φ<1˚ and the approximation t_{corr}/ t_{approx}=0.99 at the polygon boundary is rather good.

5.7.2 Errors from non-perpendicular alignment of footprint polygon

This section addresses the other two errors in the list given above. Figure 5.6 compares (for the 2D case) the projection of a footprint polygon of a voxel kernel located at (x_v, y_v). The footprint polygon has extent *2·ext* and is projected onto the projection plane, located at x_s on the viewing axis. We illustrate both the correct case, where the footprint polygon is perpendicular to the center ray, and the approximate case, in which the footprint polygon is aligned with the voxel grid. We observe that the projected polygon extent Δy_{approx} in the approximate case is slightly smaller than the projected polygon extent Δy_{corr} in the correct case.

Using simple trigonometric arguments one can show that:

$$\Delta y_{corr} = \frac{2 \cdot ext \cdot x_s (y_v \sin\varphi_c - x_v \cos\varphi_c)}{x_v^2 - ext^2 \cdot \sin\varphi_c^{\,2}} \tag{5.6}$$

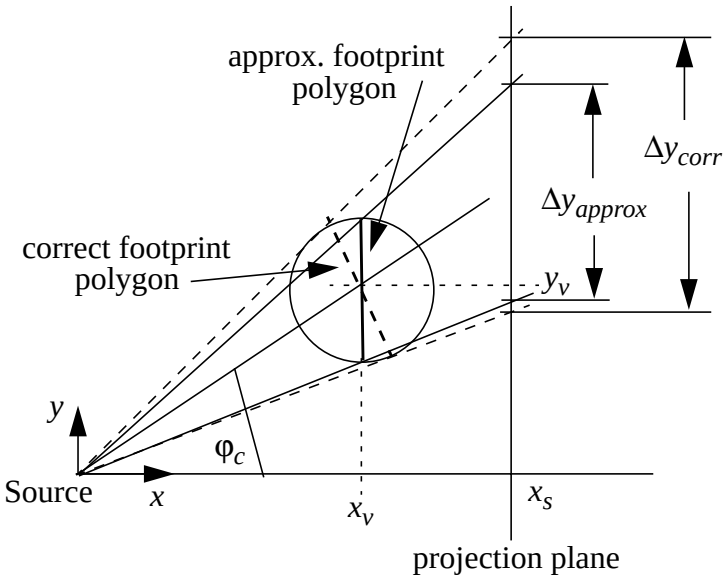


FIGURE 5.6: Errors from the non-perpendicular alignment of the footprint polygon.

One can also show that:

$$\Delta y_{approx} = \frac{2 \cdot ext \cdot x_s}{x_v} \tag{5.7}$$

Thus:

$$\Delta y_{corr} = \Delta y_{approx} \cdot \frac{x_v^2}{\cos\varphi_c\left(x_v^2 - ext^2 \cdot \sin\varphi_c{}^2\right)} \approx \Delta y_{approx} \cdot \frac{1}{\cos\varphi_c} \tag{5.8}$$

since $x_v^2 \gg ext^2 \cdot \sin\varphi_c{}^2$. This means that the error is largest at the boundary of the viewing cone. For a cone half-angle $\gamma=\varphi_c=30^\circ$, the scaling factor is 1.15. The absolute error $\Delta y_{approx} - \Delta y_{corr}$ between the correct and the approximate screen projection of a kernel with extent *ext* is given by:

$$\begin{aligned} \Delta y_{corr} - \Delta y_{approx} &= \frac{x_v^2 - \cos\varphi_c\left(x_v^2 - ext^2 \cdot \sin\varphi_c{}^2\right)}{x_v \cos\varphi_c\left(x_v^2 - ext^2 \cdot \sin\varphi_c{}^2\right)} \\ &\approx \frac{2 \cdot ext \cdot x_s\,(1-\cos\varphi_c)}{x_v \cos\varphi_c} = 2 \cdot ext \cdot \frac{x_s}{x_v}\left(\frac{1}{\cos\varphi_c} - 1\right) \end{aligned} \tag{5.9}$$

To express this error in pixels, we scale $\Delta y_{corr} - \Delta y_{approx}$ by the pixel width $\Delta_{pix}=n/w_s$. Here, n is the number of image pixels and w_s is the width of the projection plane, given by $w_s = 2x_s \tan\gamma$. This normalized absolute error is then written as:

$$\frac{\Delta y_{corr} - \Delta y_{approx}}{\Delta_{pix}} = ext \cdot \frac{n}{x_v \cdot \tan\gamma}\left(\frac{1}{\cos\varphi_c} - 1\right) \tag{5.10}$$

We observe that the error is largest for voxels close to the source and out on the cone boundary. Recall that we are only considering voxels within a spherical reconstruction region (a circle in 2D). Hence, the error is largest along the boundary of this sphere, where φ_c is given by the following expression (see also Figure 5.7):

$$\varphi_c = \text{atan}\left(\frac{\sqrt{\left(\frac{n}{2}\right)^2 - (x_{ctr} - x_v)^2}}{x_v}\right) \tag{5.11}$$

Here, x_{ctr} is the location of the volume center slice.

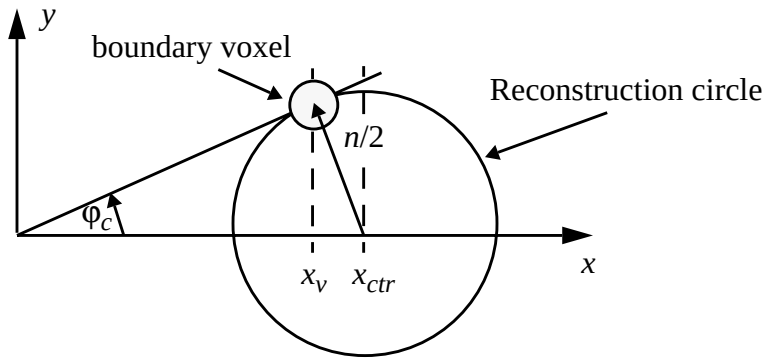


FIGURE 5.7: The angle φ_c as a function of boundary voxel location on the reconstruction circle.

Plugging the angle φ_c given by equation (5.11) into equation (5.10) yields the maximum normalized absolute error due to the non-perpendicular alignment of the footprint polygons in the context of 3D reconstruction. This error is plotted in Figure 5.8 for $x \le x_{ctr}$ (and n=128, γ=30˚, ext=2.0). We observe that the largest error is close to 0.8 pixels.

Note that these type of errors are not only committed when mapping the extent of the polygon, but also when mapping the inside of the polygon, i.e. the footprint lookup table, onto the projection plane.

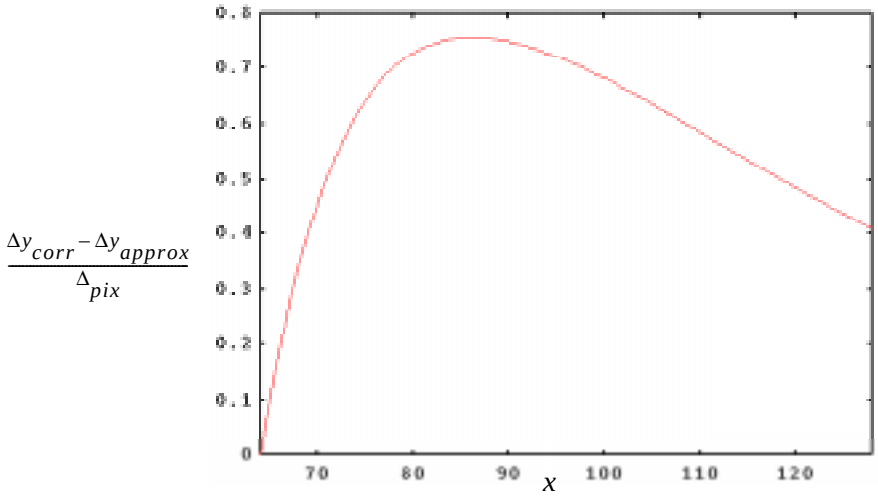


FIGURE 5.8: Maximum normalized absolute error that occurs due to non-perpendicular alignment of the footprint polygons. The error is largest for the voxels located on the boundary of the reconstruction circle (sphere in 3D). (n=128, γ=30˚, ext=2.0.)

CHAPTER 6

RAPID ART BY GRAPHICS HARDWARE ACCELERATION

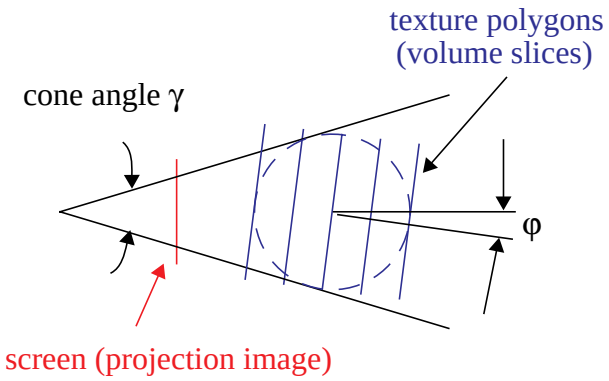
In the previous chapter, we have outlined a number of fast implementations of algebraic methods that only require the presence of a general purpose CPU. We have then contrasted the computation time of these implementations with a CPU-based implementation of the Filtered Backprojection (FBP) algorithm. This comparison revealed that both CPU-based ART and CPU-based FBP have similar runtimes. We then concluded that ART and FBP may be used interchangeably in a clinical setting, depending on the reconstruction task at hand. This conclusion, however, hides some of the facts of clinical reality. Today's CT scanners do not use general purpose CPUs, rather, they always incorporate custom DSP (Digital Signal Processing) chips which run the FBP algorithm extremely fast. At first glance, we may take this reality-check as a hint that our cost analysis of Section 5.6 has only theoretical merit. However, that is not quite true either. The cost analysis of Section 5.6 compares the runtime complexities of the two algorithm, and we have shown that these runtime complexities are about equal. So if we took ART and built custom hardware to run it on, we would expect it to run just as fast as the accelerated FBP algorithms. However, custom hardware is expensive to build and it limits the scope at which an algorithm can be used. Once the hardware is in place, modifications and adaptations are difficult to make, which hampers the evolution of a technology. Acknowledging all these drawbacks, we have taken a different approach. Instead of designing special chips to implement our ART and SART algorithms, we have chosen widely available graphics workstations as our acceleration medium. Great prospects can be expected from this choice, as the graphics hardware resident in these workstations is especially designed for fast projection operations, the main ingredients of the algebraic algorithms. Another plus of this choice is the growing availability of these machines in hospitals, where they are more and more utilized in the daily task of medical visualization, diagnosis, and surgical planning. The feature of these graphics workstation we will rely on most is *texture mapping*. This technique is commonly used to enhance the realism of the polygonal graphics objects by painting pictures onto them prior to display. Texture mapping is not always, but often, implemented in hardware,

and runs at fill rates of over 100 Megapixels/sec. However, hardware texture mapping is not limited to graphics workstations only, many manufacturers offer texture-mapping boards that can be added to any modern PC.

In the following section, we will describe a *T*exture-*M*apping hardware *A*ccelerated version of ART (TMA-ART) that reconstructs a 128^3 volume from 80 projections in less than 2 minutes — a speedup of over 75 considering the software solution outlined in Chapter 5. The programs were written using the widely accepted OpenGL API (Application Programming Interface).

6.1 Hardware Accelerated Projection/Backprojection

TMA-ART decomposes the volume into slices and treats each slice separately. In grid projection (shown in Figure 6.1), each slice is associated with a square polygon with the volumetric slice content texture-mapped onto it. A projection image is then obtained by accumulatively rendering each such polygon into the framebuffer.



Projection Algorithm

Rotate texture polygons by projection angle φ

Texture map the volume slices onto texture polygons

Project textured polygons onto the screen

Accumulate screen contributions at each pixel

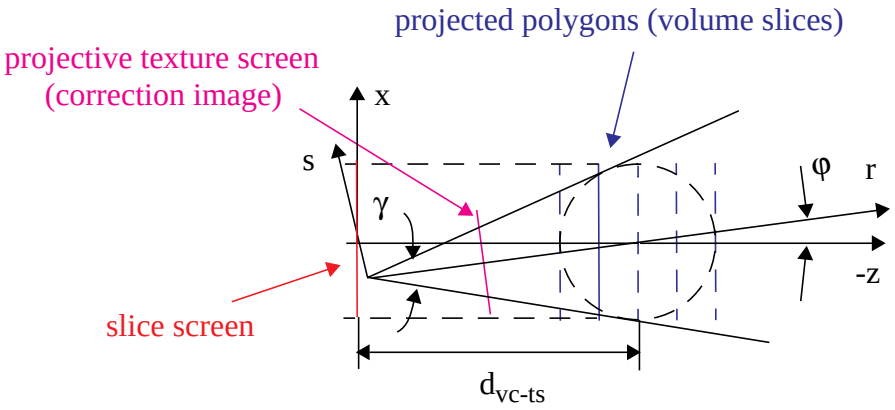
FIGURE 6.1: Grid projection with TMA-ART.

The ray integrals so computed are equivalent to the ray integrals obtained in a software solution that uses a trilinear interpolation filter and samples only within each volume slice. Note that since the distance between sample points is not identical for every ray (due to the perspective distortion), we have to normalize the projection image for this varied distance.

This can be conveniently achieved by normalizing the scanner images by the inverse amount in a pre-processing step.

After a projection image has been generated, a correction image is computed by subtracting this projection image from the scanner image. Hence, due to this image-based approach, we are using SART and not ART. Not only is this necessitated by the image-based projection approach of the graphics hardware, it is also convenient with regards to SART's insensitivity to the cone-beam related reconstruction artifacts, as discussed in Section 4.2.

In backprojection (shown in Figure 6.2), we need to distribute a correction image onto the volume slices. This is achieved by associating each volume slice with the framebuffer onto which the correction image, mapped to a polygon, is rendered. (This is the inverse situation of Figure 6.1.) Although projection is simple, backprojection is not as straightforward, since here the main viewing direction is not always perpendicular to the screen. This, how-



T_1: translate by $d_{vc\text{-}ts}$ — **R**: rotate by φ

T_1^{-1}: translate by $-d_{vc\text{-}ts}$ — **P**: perspective mapping

S: scale by 0.5 — **T_3**: translate by 0.5

Backprojection algorithm

Set texture matrix to $\mathbf{TM}=\mathbf{T_1}\cdot\mathbf{R}\cdot\mathbf{T_1^{-1}}\cdot\mathbf{P}\cdot\mathbf{S}\cdot\mathbf{T_3}$

For each volume slice

Associate 3D-texture coordinates with each vertex,

set r-coordinate to z-coordinate

Render the polygon

Use **TM** to map texture coords. onto texture screen

Add the rendered slice image to the volume slice

FIGURE 6.2: Backprojection with TMA-ART.

ever, is required by the graphics hardware. To enable this viewing geometry, we implemented a virtual slide-projector (using the projective texture approach of Segal et. al. [56]) that shines the correction image at the oblique projection angle onto a polygon, which in turn is orthographically viewed by the framebuffer. This is shown in Figure 6.2 for one representative volume slice. The correction image is perspectively mapped, according to the cone-geometry, onto the volume slice that has been placed at the appropriate position in the volume. This "slide" projection is then viewed by the screen.

Let me explain this slide-projector approach in some more detail. In OpenGL, a polygon is represented by three or more vertices. When the polygon is projected onto the screen, the coordinates of its vertices are transformed by a sequence of matrix operations, as shown in Figure 6.3. (For more detail on these fundamental issues refer to [46] and [13].)

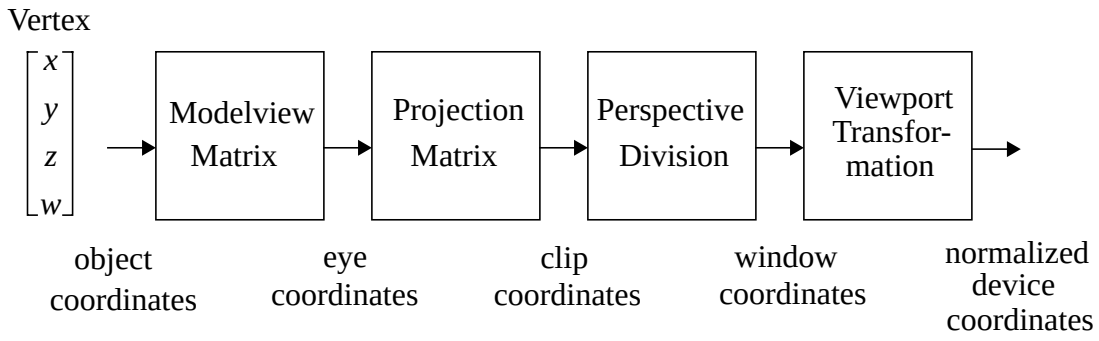


FIGURE 6.3: Stages of vertex transformation

A texture is an image indexed by coordinates in the range [0.0..1.0, 0.0...1.0]. When a texture is mapped onto a polygon, the polygon's vertices are associated with texture coordinates [s,t], as shown in Figure 6.4. The viewing transformation of the polygon vertices yields a closed region on the screen. In a process called *scan conversion*, all pixels inside this region are assigned (via interpolation) texture coordinates within the range assigned to the bounding vertices. Note that this transformation can lead to a stretching or shrinking of the texture.

The texture mapping coordinates need not be two-dimensional. As a matter of fact, they can be up to four-dimensional (involving a homogeneous coordinate), just like the vertex coordinates. In addition, OpenGL provides a transformation facility, similar to the one supplied for vertex transformation, with which the interpolated texture coordinates can be transformed prior to indexing the texture image. We can use this facility to implement our virtual slice projector.

The algorithm proceeds as follows. First, we create an array of n square texture coordinate polygons with vertex coordinates (s,t,r). Here, we set the (s,t) coordinates to (n, n), i.e., the extent of the volume slices. The r-coordinate we vary between [$d_{\text{vc-ts}}$-n/2, $d_{\text{vc-ts}}$+n/2]. ($d_{\text{vc-ts}}$ is the distance of the source to the volume center.) Refer now back to Figure 6.2,

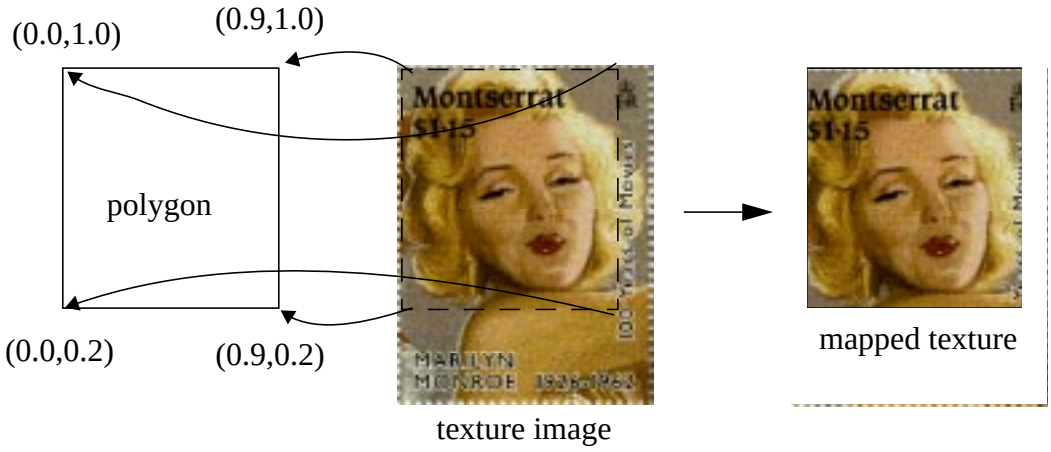


FIGURE 6.4: Texture mapping an image onto a polygon. The texture coordinates assigned to the polygon vertices are given in parenthesis.

where we show the decomposition of the texture transformation matrix. The Modelview matrix is set to the product $\mathbf{T_1}\cdot\mathbf{R}\cdot\mathbf{T_1}^{-1}$, i.e., each polygon is rotated about the volume center by the viewing angle φ. The Projection matrix is set to a perspective mapping of the texture coordinates onto the projective texture screen. After the perspective divide, the texture coordinates would be in the range [-1.0...1.0]. Since we can only index the texture image within a range of [0.0..,1.0], we need to scale and translate the texture perspective texture coordinates prior to indexing. This is achieved by incorporating a scale and translation given by $\mathbf{S}\cdot\mathbf{T_3}$ into the Projection matrix.

We can now perform the backprojection of the correction image, represented by the texture, onto the volume slices. Let us just look at one of the volume slices, represented by polygon P_s with vertex coordinates (n, n, z), which is projected orthographically on the slice screen. Depending on its z-location, the polygon is assigned one of the texture coordinate polygons. When mapping P_s onto the screen, texture coordinates are generated for each pixel within the projected polygon extent. However, these texture coordinates are not used directly to index the correction image, but are first passed through the texture transformation pipeline. The transformed coordinates then index the correction image texture as if this image had been projected onto P_s at the backprojection angle φ.

One should add that this process is not any more expensive than direct texture mapping. Once the texture transformation matrix is compounded, just one hardware vector-matrix multiplication is needed. As a matter of fact, this multiplication is always performed even if the texture transformation is unity.

Utilizing the texture mapping hardware for the grid projection and backprojection operations resulted in a considerable speedup: a cone-beam reconstruction of a 128^3 volume from 80 projections could now be performed in 15 minutes, down from the 2.5 hours that were required in the software implementation. These were certainly encouraging results,

which prompted us to seek further avenues for hardware acceleration. For this purpose we dissected the program flow in its main constituents, as reported in the next section.

6.2 Potential for Other Hardware Acceleration

Consider Figure 6.5 for a detailed analysis of the program flow (compare also Figure 2.6). The portions of the program that use hardware acceleration are highlighted. We notice that a substantial part of the program deals with transferring data to and from the graphics engine, and also to and from main memory. We also notice that there are three steps in the program that have potential for hardware acceleration. These portions are:

- The accumulation of the projection image (in the projection step).
- The computation of the correction image.
- The volume update, i.e., the addition of the rendered correction slice to the respective volume slice in main memory (in the backprojection step).

Before we consider each of these items in detail, let us first introduce the main datastructures used in the program. They are listed in Table 6.1.

Data structure	ART value range	TMA-ART value range
volume array	[0.0...1.0]	[0.0...1.0]
weight image	[0.0...*n*]	[0.0...*n*]
scanner image	[0.0...*n*]	[0.0...*n*]
projected slice image	[0.0...1.0]	[0.0...1.0]
full projection image	[0.0...*n*]	[0.0...*n*]
correction image	[-1.0...1.0]	[0.0...1.0]
backprojecton image	[0.0...1.0]	[0.0...1.0]

TABLE 6.1 Data structures of ART and TMA-ART and their pixel value ranges.

The graphics hardware buffers and texture memory can only hold values in the range [0.0...1.0]. If these values are exceeded, saturation (clamping) occurs. Therefore, all data structures that use the graphics pipeline must have their values scaled to this range. Since we project the volume slices via the graphics hardware, we cannot have volume values outside the range [0.0...1.0]. The ranges of all other data structures follow from that constraint. If the volume is the range [0.0...1.0], the projections will necessarily be in the range [0.0...*n*]. Most likely, the scanner images will not be within this interval, but a simple scal-

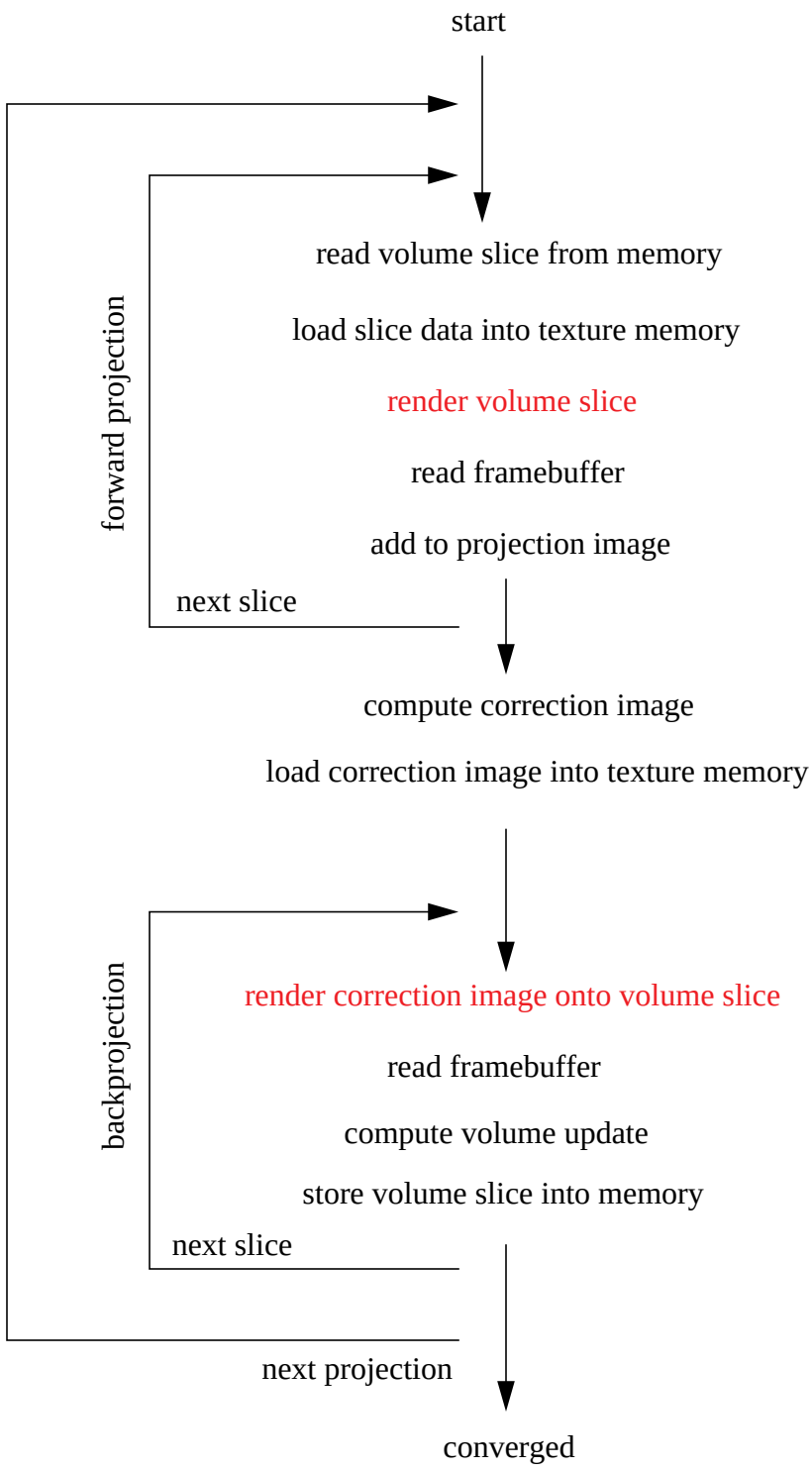


FIGURE 6.5: Flow chart of TMA-ART.

ing will get them there.

The backprojected correction image is a special case. Since the volume values are within [0.0...1.0], the corrections must be within [-1.0...1.0]. Obviously, we cannot render this image directly, since the graphics hardware only supports values in [0.0...1.0]. Instead, we scale the values by the following expression:

$$corrImg_{TMA-ART} = \frac{corrImg_{ART} + 1.0}{2.0} \tag{6.1}$$

The scaled values are then written to the texture memory and projected onto the back-projection images using the slide projector algorithm described before. The rendered back-projections must then be scaled back prior to adding them to the respective volume slices:

$$backprojImg_{ART} = backprojImg_{TMA-ART} \cdot 2.0 - 1.0 \tag{6.2}$$

Recall from equation (2.17) that we need to normalize the correction images by sums of weights. These can sums of weights can be conveniently represented by *M* weight images, one for each projection angle φ, which can be pre-computed by projecting a volume that is unity everywhere within the reconstruction sphere.

Let us now consider the three candidates for hardware acceleration.

6.2.1 Accumulation of the projection image

In most graphics workstations that come equipped with texture mapping hardware, the framebuffer can hold up to 12 bits, although there are some low-end machines that offer only 8 bits. The texture memory usually matches the resolution of the framebuffer. Our base implementation accumulates the projected volume slices in a software buffer. This requires a (relatively expensive) framebuffer read for each projected volume slice. We could decrease the number of framebuffer reads by performing some of the accumulations directly in the framebuffer. This comes at a price, however, since in order to accomplish these accumulations, we must sacrifice framebuffer precision bits. Obviously, this would not be wise as it would decrease the accuracy of the rendered projection images.

There is, however, a trick that we can use. The framebuffer has three color channels, Red, Green, Blue, and an Alpha channel. Usually, we are only reconstructing grey level data, so all we utilize is a single color channel, say Red, both in texture memory and in the framebuffer. This is shown in Figure 6.6.

However, if we partition the 12bit data word into two components, one 8 bit and one 4 bit, and render it into two separate color channels, Red and Green, then we can accumulate data into the remaining upper 4 bits of the two framebuffer channels. This is illustrated in Figure 6.7.

These 4 bits allow us to accumulate up to 16 images, which decreases the number of necessary framebuffer reads by 2/16 (we now have to read two color channels). Notice, however, that $bit_{8\text{-}11}$of the texture word are not interpolated by the texture mapping hardware

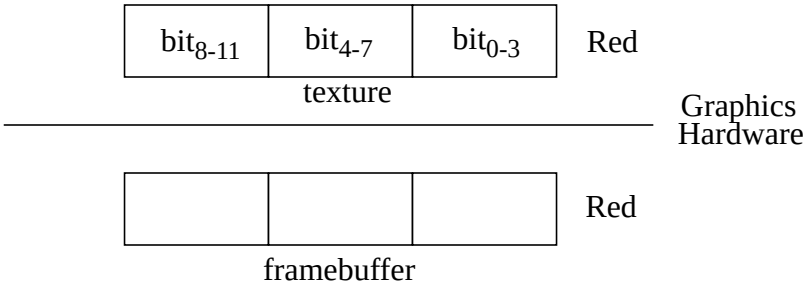


FIGURE 6.6: Rendering a full 12bit data word into a 12bit framebuffer. The full range of the framebuffer is exploited, no accumulations can be made.

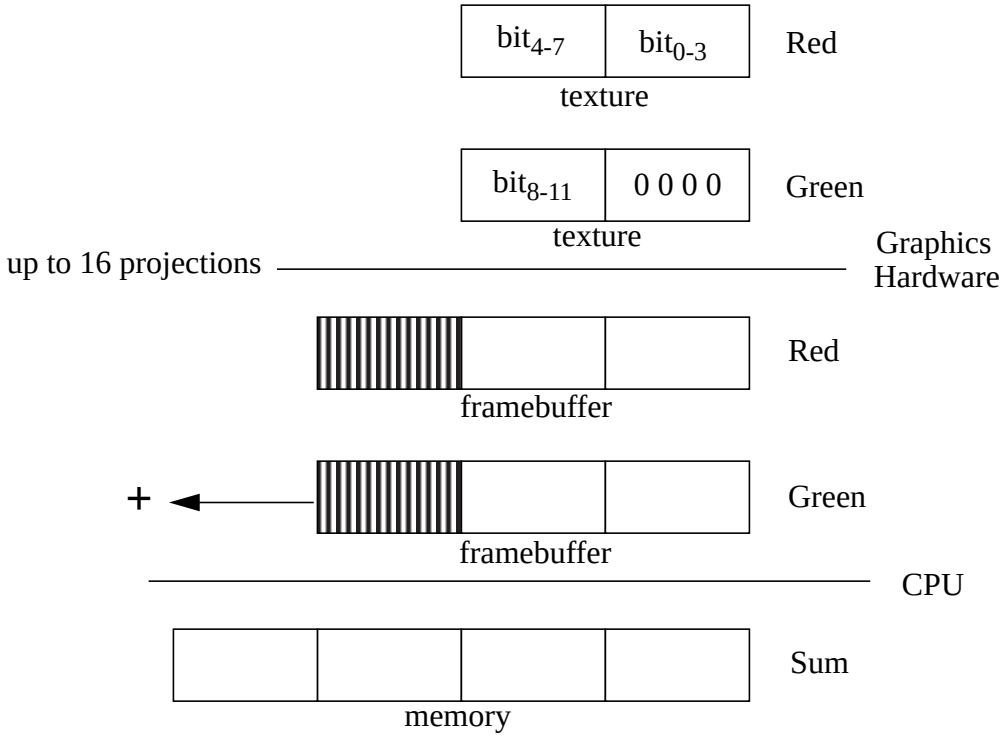


FIGURE 6.7: Rendering a 12bit data word using 2 color channels. The shaded upper 4 bits in the framebuffer can be used for accumulation. After the 4 upper bits have been filled by 16 projections, we must add the two channels in the CPU. For this purpose, the green channel must be shifted to the left by 4 bits.

in 12 bits, but only in 8 bits. This may cause inaccuracies. To illustrate this problem, image the following simple case. Assume a *texel* (texture element) has a binary value of 1,0000,0000 (only bit_8 is set) and its immediate neighbors are all 0. Thus the red texture channel contains 0, and the green texture channel contains 1,0000. Now let's say that the

texture mapping interpolation of this texel neighborhood yields a binary value of 1000. In the first approach (Figure 6.6), the framebuffer would contain that value, in the second approach (Figure 6.7), however, the framebuffer would contain 0.

Finally, we should note that there exists graphics hardware that offers a dedicated hardware accumulation buffer [20]. This hardware, however, is relatively rare and also rather expensive.

6.2.2 Computation of the correction image

A correction image is computed as follows (incorporating our earlier comments about scaling):

$$\begin{aligned} corrImg_{TMA-ART} &= \left(\frac{scannerImg - projImg}{weightImg} \right) \cdot 0.5 + 0.5 \\ &= \frac{corrImg_{ART}}{2.0} + 1.0 \end{aligned} \tag{6.3}$$

The associated value ranges are as follows:

$$\begin{aligned} [0.0 \ldots 1.0] &= \left(\frac{[0.0 \ldots n] - [0.0 \ldots n]}{[0.0 \ldots n]} \right) \cdot 0.5 + 0.5 \\ &= \frac{[-1.0 \ldots 1.0]}{2.0} + 1.0 \end{aligned} \tag{6.4}$$

We see that, due to the excessive value ranges, $corrImg_{ART}$ cannot be computed in hardware. The scaling cannot be done in hardware either, due to the extended range [-1.0...1.0] of $corrImg_{ART}$.

6.2.3 Volume update

Our basic TMA-ART implementation performs the volume update on the CPU. But maybe this can be done using the texture mapping facility. Let us look at the equation involved:

$$volSlice = \left\lfloor \left\lceil volSlice + backprojImg_{TMA-ART} \cdot 2.0 - 1.0 \right\rceil_{1.0} \right\rfloor_{0.0} \tag{6.5}$$

The ceiling and floor operations bound the volume density values to a range [0.0...1.0]. This is in line with general practices in algebraic methods, as the original object did not occupy these density values either.

The associated ranges are as follows:

$$\begin{aligned} [0 \ldots 1.0] &= \left\lfloor \left\lceil [0 \ldots 1.0] + [0 \ldots 1.0] \cdot 2.0 - 1.0 \right\rceil_{1.0} \right\rfloor_{0.0} \\ &= \left\lfloor \left\lceil [0 \ldots 1.0] + [0 \ldots 2.0] - 1.0 \right\rceil_{1.0} \right\rfloor_{0.0} \\ &= \left\lfloor \left\lceil [0 \ldots 1.0] + [-1.0 \ldots 1.0] \right\rceil_{1.0} \right\rfloor_{0.0} \end{aligned} \tag{6.6}$$

Due to the excessive ranges of the values, updating of the volume is not easily done in hardware. However, we can think of a more elaborate scheme that updates the volume in two stages. First, it is updated with the negative part of the correction, then it is updated with the positive part. This is illustrated in Figure 6.8.

Equation: $vol = vol + 2.0 \cdot corr - 1.0 = vol - 2.0 \cdot corr^{-} + 2.0 \cdot corr^{+}$

Hardware implementation:

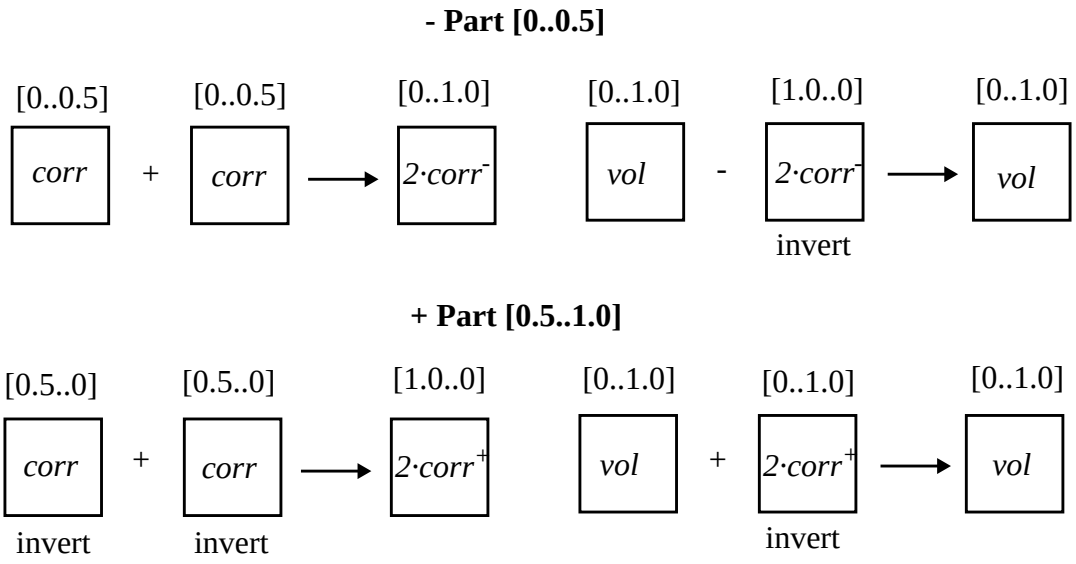


FIGURE 6.8: Hardware implementation of the volume update. The boxes denote images and the intervals printed above the boxes denote the intervals of the image that are used for blending. The *corr* image is the backprojection image that was rendered using the slide projector and now resides in the front framebuffer. The *vol* image is the volume slice to be updated.

By using the OpenGL function *glAlphaFunc()* one can limit the value interval of the pixels that is accepted by the framebuffer. Values outside the specified interval are not written to the framebuffer. Using *glAlphaFunc()* mechanism allows us to specify pixel intervals other then the default [0.0...1.0]. OpenGL also provides a function *glLogicOp()* that can be used to complement a pixel value. This function is employed to invert the images where specified in the diagram of Figure 6.8. Note that all operations can be performed by copying images back and forth within the several GL buffers that are available (such as front and back buffers, auxiliary buffers, and pBuffers). Thus expensive framebuffer reads and texture memory loads can be avoided. Recall that the *corr* image is the backprojection image that was rendered using the slide projector and already resides in the front framebuffer. Thus, updating a volume slice with the correction image takes one polygon projection and

four framebuffer-based additions.

We now turn to the other main constituent of Figure 6.5: the transfer of data to and from main memory.

6.3 Optimal Memory Access

The volume data is stored as 16 bit unsigned shorts, while the projection and weight image data are stored as 32 bit unsigned integers, since they represent accumulations of many 16 bit voxel slices. The correction image is stored in 16 bit, since it is normalized to one volume slice prior to backprojection. Thus the precision of the reconstruction process is inherently 16 bit, hampered, however, by the limited 12 bit resolution of the framebuffer. We will look into a virtual bit extension of the framebuffer later.

Consider Figure 6.9 where we illustrate the three different viewing directions (VDs) at which the memory is accessed during the reconstruction.

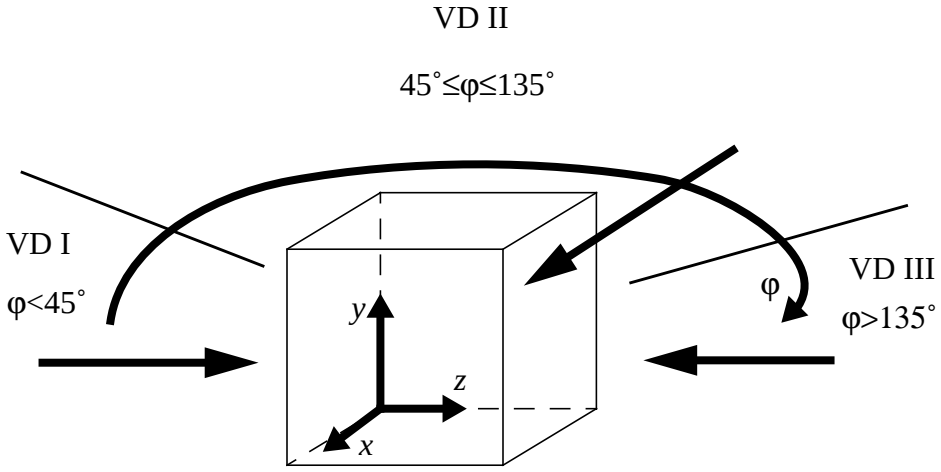


FIGURE 6.9: Memory access order for different projection angles φ. There are three major viewing directions (VDs). For VD I and VD III the volume data are accessed one *z*-slice at a time, while for VD II, the volume data are accessed one *y*-slice at a time.

Usually, computers organize their memory in a hierarchy of several levels: a small primary on-chip cache, a larger secondary on-chip cache, main memory, and some swap partition on disk. Whenever a datum requested from the CPU is not found in one level, a cache or memory fault is generated and the datum is fetched from the next lower memory level, and so on. When the datum has reached the CPU, it is also stored in all memory levels that were traversed on the quest. Usually, the data are fetched in blocks, following the law of locality-of-reference which states that once a datum is requested it is likely that the neighboring datum will be needed soon. Since memory access time is generally much larger than trans-

fer time, it is thus more efficient to transfer a whole block once the memory access is performed, than to access every datum anew. Therefore, once a block of data is loaded into cache, no more faults are generated for any data in the block. Fetching data from the caches is generally much faster than retrieving the data from main memory. In case of a secondary-cache fault, the CPU will in many cases switch to another job until the data block is in, while at a primary cache-fault the CPU will just introduce a few wait states. Hence, we want to keep the number of secondary cache faults as low as possible. (For more details on memory hierarchies please refer to [58])

Our SGI Octane workstation with a Mips R10000 CPU has a secondary cache of 1MByte, with 128-byte slots and 2-way set-associativity. The secondary cache has a peak transfer rate of 3.2GByte/s to the CPU, which is magnitudes higher than the transfer rate from main memory. Since each volume voxel occupies 2 bytes, we can store 64 voxels in each cache slot.

We need to be intelligent about how we store the data in memory. The minimum number of cache faults is $N/64$ (where N is the number of voxels). Consider Figure 6.10 where we show the case in which the data are stored in *x*-*y*-*z* order, i.e., *x* runs fastest. If we access the data from VD I or VD III in *x*-*y* order for each slice, we will encounter a cache fault every 64 voxels, the minimum cache-fault rate. However, if we access the data from VD II in *z*-*y* order (or *y*-*z* order) for each subsequent *x*-slice, we will have a cache fault for every voxel. This is 64 times more cache faults than for VD I and VD III, and will slow the reconstruction down considerably. Note also that these cache faults may occur not only for the loading of the volume slices, but also for the storing of the corrected volume slices, if a write-back cache-policy is used.

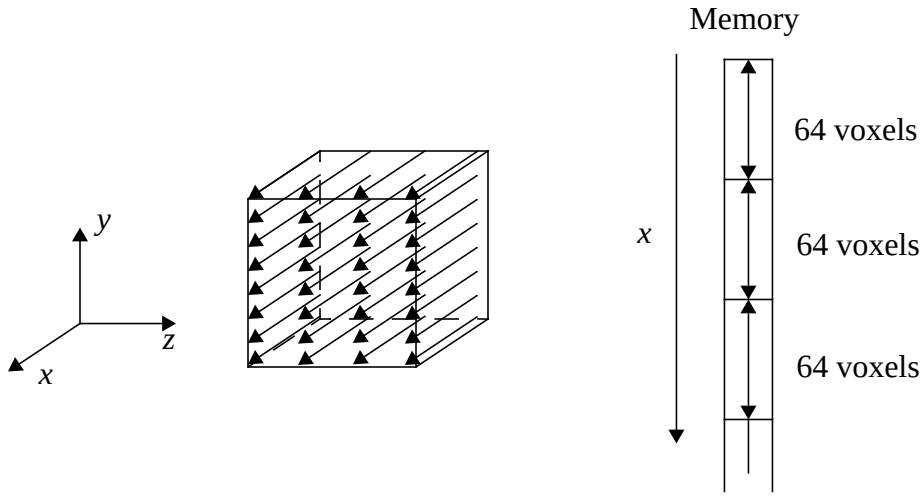


FIGURE 6.10: Voxels are stored in *x*-*y*-*z* order. No extra cache faults are generated for VD I and VD III. For VD II, however, a cache fault is generated for every voxel.

Consider now Figure 6.10 where we show a different, more favorable memory organization in which the voxels are stored in *y*-*x*-*z* order. Let us look at VD I and VD III first. If we access the data in *y*-*x* order for every *z*-slice, then we generate a cache fault every 64 voxels, just like before. On the other hand, for VD II, if we access the data in *y*-*z* order for every *x*-slice, then we will not have any extra cache faults either. Thus the *y*-*x*-*z* storage arrangement in conjunction with *y*-first memory traversal will yield the optimal cache behavior.

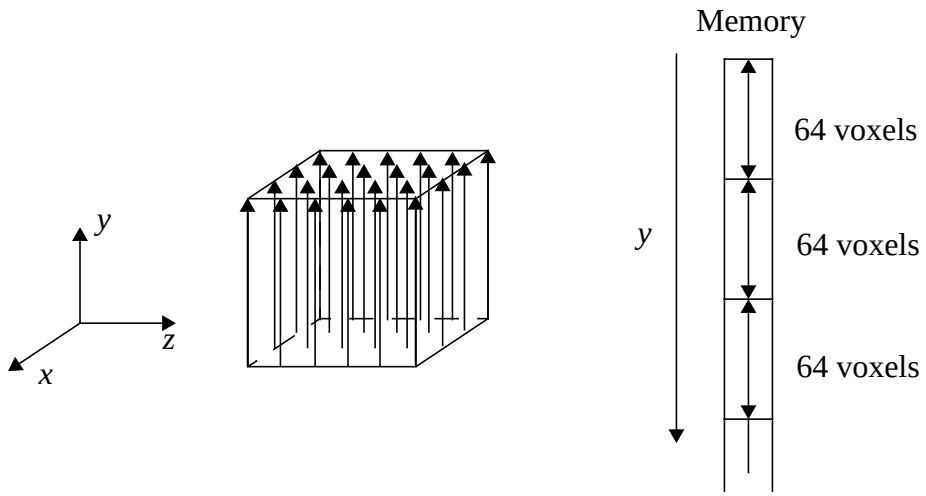


FIGURE 6.11: Voxels are stored in *y*-*x*-*z* order. No extra cache faults are generated for VD I, VD II and VD III.

The following example may serve as a demonstration for the importance of optimizing cache behavior. In our initial implementation, we used the most natural storage scheme, the *x*-*y*-*z* order, with which a reconstruction could be performed in 15 minutes. After observing the large number of cache faults (using SGI's *perfex* utility), we switched to the *y*-*x*-*z* storage order and the *y*-first data traversal. Doing so enabled a reconstruction to be completed in a mere 2 minutes — a speedup of 7.5.

6.4 Increasing the Resolution of the Framebuffer

We have mentioned before that all data are stored in 16 bit precision, although the framebuffer is only capable of generating the projection images in 12 bit resolution. Hence, 4 bits of resolution are practically wasted — a lack of precision that could be crucial for recovering small detail in the reconstruction. In this section, we shall explore a scheme that conquers the remaining 4 bits of resolution from the framebuffer.

Recall Section 6.2.1 where we showed how the width of the framebuffer can be extended to implement an accumulation buffer. A similar scheme can be used for enhancing the

framebuffer's precision. The two schemes can also be combined to yield a high precision accumulation buffer.

6.4.1 Enhancing the framebuffer precision

Consider Figure 6.12 where these concepts are illustrated. The lower 12 bits of the volume data (or correction image data) are written to the Red texture channel, while the upper 4 bits of the data are written to the upper 4 bits of the Green texture channel. Rendering is performed as usual, and the Red and the Green framebuffer is read into two software buffers. The 16 bit result is constructed by adding the two software buffers, with the data in the Green buffer shifted to the left by 4 bits.

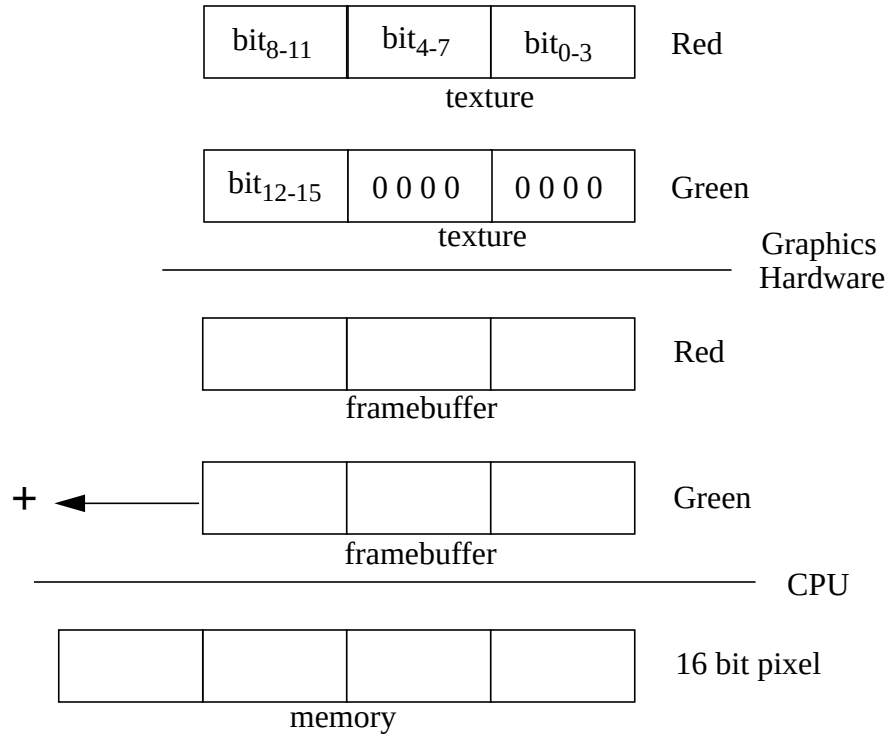


FIGURE 6.12: Increasing the framebuffer resolution from 12 bit to 16 bit by adding up two color channels, properly shifted.

Note that, similar to the accumulation buffer, the (virtual) 16 bit data in the Green channel are not interpolated in 16 bits, but only in 12 bits. This will have effects similar to the ones outlined in Section 6.4.1. Hence, the presented implementation is not a true 16 bit extension to the framebuffer, it is only an approximation. However, it will still produce considerably more precise results than the original 12 bit implementation.

6.4.2 A high precision accumulation framebuffer

We can combine the concepts of Section 6.4.1 (high precision framebuffer) and Section 6.2.1 (accumulation framebuffer) to create a high precision accumulation framebuffer used in the projection portion of TMA-ART. This is shown in Figure 6.13. The result is a 20 bit sum that is an accumulation of up to 16 16-bit texture projections. This scheme is similar to the one proposed by Shochet [57]. It reduces the number of framebuffer reads to 3/32 of the non-accumulative high precision implementation. Note that both the words in the Green and the Blue channel are not interpolated to the full extent (see our comments above).

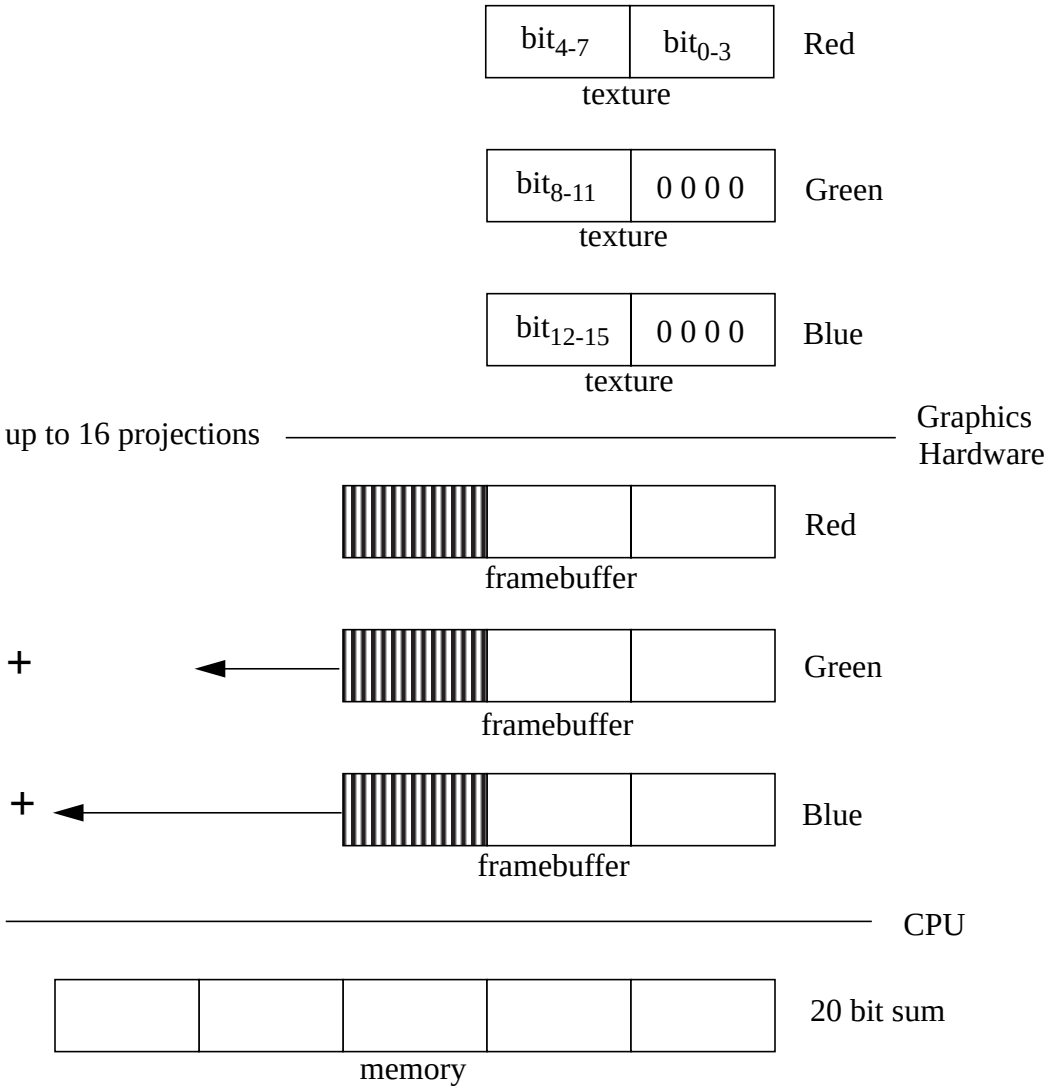


FIGURE 6.13: Accumulation framebuffer with 16 bit precision. The shaded areas denote the bits used for accumulation.

6.5 Results

Table 6.2 presents the time effort spent for each of the program portions (tasks) listed in the

task	compounded time (sec)	percentage/total
read volume slices	4.6	10.9
load volume slices into texture memory	4.3	10.1
render volume slices	0.2	0.5
read framebuffer	5.6	13.3
accumulate projection image	4.7	11.1
compute projection image	0.3	0.7
load correction image into texture memory	0.04	0.1
render correction image onto volume slices	0.1	0.2
read framebuffer	6.3	15.0
compute volume update	12.0	28.4
store volume slices into memory	4.1	1.0
total time for 1 iteration	42.2	100.0

TABLE 6.2 Time performance (in seconds) of cone-beam TMA-ART for the different constituents (tasks) of the program flow chart given in Figure 6.5. Each number represents the wall-clock time, compounded over one iteration (80 projections of 127^2 pixels each, reconstructing on a 128^3 grid). The percentage of the computational effort of each task with respect to the total time is also given. The timing were obtained on a SGI Octane with a R10000 CPU.

flow chart of Figure 6.5. The values shown represent the wall-clock time compounded over one iteration of cone-beam TMA-ART, reconstructing from 80 projections on a 128^3 grid. The timings are given for the basic implementation using the *y*-*x*-*z* volume storage and the *y*-first traversal scheme, but none of the other enhancements discussed. We notice that the time spent on image projection is insignificant compared to the time required for loads and stores, and the time spent on CPU tasks, such as for projection accumulation and the computation of the volume update.

Table 6.3 lists the runtime of TMA-ART when the enhancements described previously in this chapter are added. We observe that by adding the virtual accumulation buffer to reduce

the amount of expensive framebuffer reads in the projection phase of ART, we can reduce the runtimes considerably. We can even compensate for the cost of the 16 bit extension of the framebuffer.

accumulation buffer	16 bit framebuffer	time / iteration (sec)	reconstruction time (sec)	added cost (%)
-	-	42.2	126.6	-
✓	-	33.18	99.54	-21
-	✓	62.0	186.0	+32
✓	✓	37.2	111.6	-12

TABLE 6.3 Runtimes per iteration for different enhancements of TMA-ART. Timings refer to the conditions listed in the caption of Table 6.2. A reconstruction takes 3 iterations.

Let us now discuss TMA-ART in terms of reconstruction quality. Consider Figure 6.14 where several reconstructions of the 3D Shepp-Logan phantom are illustrated. On the left we show three reconstructions obtained with the basic 12-bit framebuffer TMA-ART, while on the right we show three reconstructions obtained with the enhanced 16-bit framebuffer TMA-ART. The contrast of the imaged phantom doubles in every row from top to bottom (see Table 4.1 for the phantom definition). We observe that the basic TMA-ART produces reconstructions with consistently higher noise levels than the 16 bit TMA-ART. The difference is particularly striking in the first row for the original contrast. Here, the three small tumors in the lower third of the slice are only discernible in the 16-bit TMA-ART reconstruction. However, the higher the contrast, the lesser the impact of the framebuffer precision. The reconstructions obtained at twice the contrast of the Shepp-Logan phantom are already of acceptable quality.

It is now clear that with 16 bit framebuffer resolution the quality of the software implementation cannot be reached. However, it is hoped that the next generation of graphics workstations will add 4 extra bits to the framebuffer, which would then enable a virtual 20 bit framebuffer extension. It is expected that this will bring the reconstruction quality closer to the maximum.

Since I did not have a clinical cone-beam projection dataset available to me at the time of submission of this dissertation, I could not assess the fitness of cone-beam TMA-ART for real clinical applications. However, the fact that cone-beam TMA-ART reconstructs the 3D Shepp-Logan phantom — a widely accepted benchmark — rather well, leaves no doubt that the algorithm has great potential to succeed in the clinical world. The reconstruction speed of less than two minutes for a complete reconstruction of a 128^3 dataset is competitive, even in comparison with dedicated hardware boards.

6.6 Future Work - Parallel Implementations

I shall close with another look at the flow chart of Figure 6.5. We realize that none of the large blocks in this flow chart can be interleaved. Framebuffer reads and texture memory writes involve the CPU, and the rendering time is minimal compared to all other costs. However, one can easily parallelize the projection image accumulation in the projection phase and the volume update computation in the backprojection phase. One could think of a scheme in which one or more CPUs accumulate the projection image/do the volume update computation (basically a multiplication, an addition, and two clamping operations per voxel), while another CPU feeds and retrieves the graphics data. Then the graphics-related CPU operations (with only one graphics engine available) are parallel to the pure CPU-related operation. Looking at the timings of Table 6.2, good balance may already be achieved with two processors in the projection phase and three processors in the back-projection phase. We plan to investigate this concept further in the future.

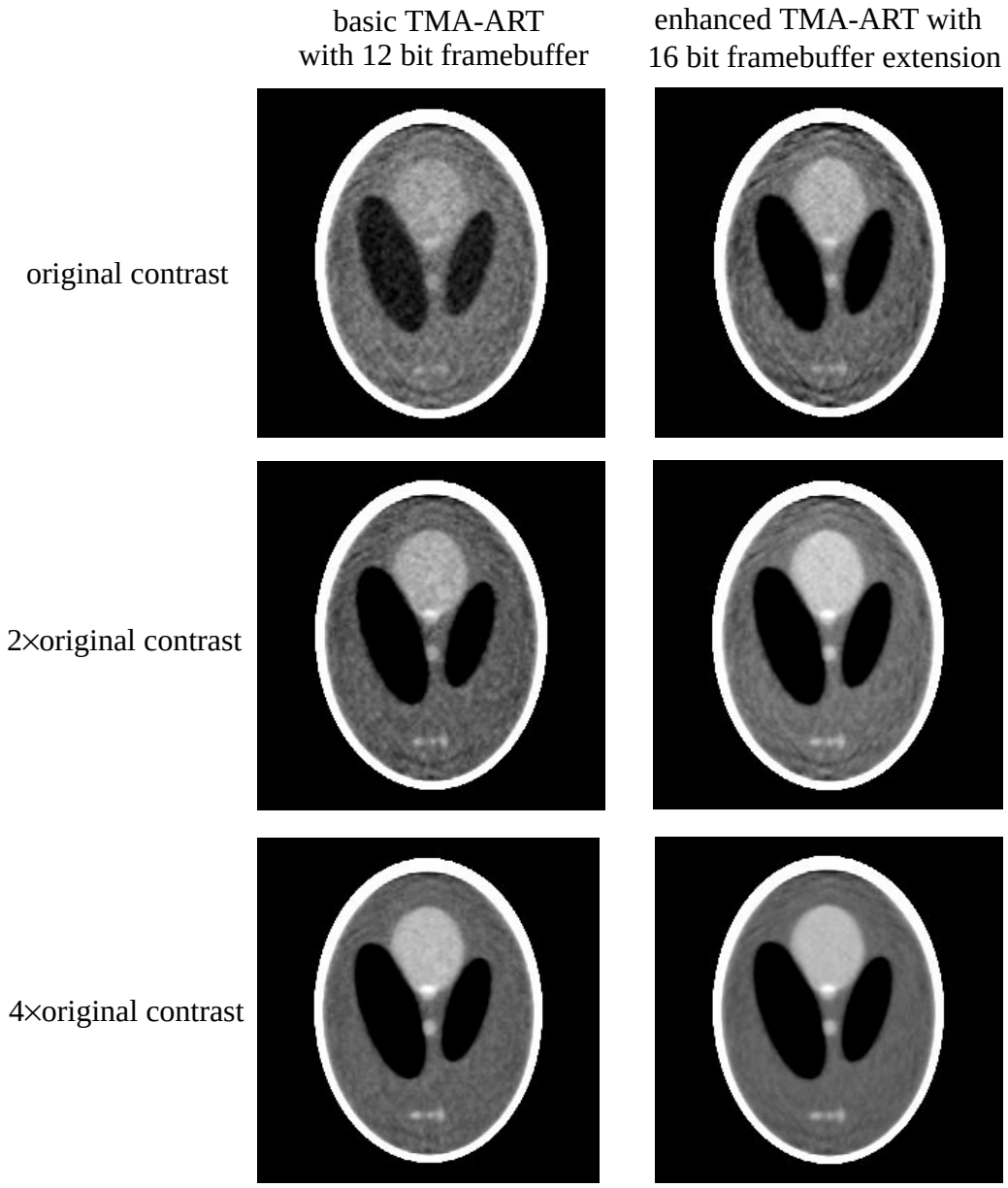


FIGURE 6.14: 3D Shepp-Logan brain phantom reconstructed with both basic TMA-ART and precision-enhanced TMA-ART. Contrast was varied from the original contrast to 4 times the original contrast. Notice that the precision-enhanced TMA-ART consistently reconstructs the phantom with less noise artifacts than the basic TMA-ART. (128^3 grid, 80 projection, cone angle γ=40˚, and 3 iterations.)

CHAPTER 7

CONCLUSIONS

The contributions of this dissertation, along with their corresponding publications, are as follows:

- Analysis of the accuracy and efficiency of various existing projection algorithms in the context of cone-beam projection. ([37])
- Extension of a well known projection algorithm, i.e., splatting [66]-[68], from orthographic to perspective (cone-beam) projection. Although perspective splatting has been proposed before (by the original author), the new extension is considerable more accurate and also optimized for speed. ([37][39][41])
- Cost analysis of object-order (i.e., splatting) vs. image-order projection algorithms in the context of perspective projection. This analysis revealed that image-driven algorithms are inherently better suited for fast perspective projection. Hence, a 3D extension of an existing 2D image-order projection algorithm [22] was proposed (and termed ray-driven splatting) and optimized for speed. ([37][39][41])
- Application of the new perspective ray-driven splatter as the projection engine in iterative 3D cone-beam tomographic reconstruction with algebraic methods. While these algebraic methods are well-known for their use in parallel-beam and 2D fan-beam reconstruction, their extension for low-contrast, cone-beam reconstruction is new. ([37])
- Extension of the most common algebraic method, the Algebraic Reconstruction Technique (ART), to properly handle the non-uniform grid sampling situation of perspective rays. The undersampling in some grid regions causes ART, in the present line-integral configuration, to generate reconstructions with considerable aliasing artifacts. Various new solutions to alleviate these problems were proposed: 1. Pixel supersampling; 2. Pyramidal rays in conjunction with summed-area tables; 3. Increasing the size of the splatting kernels as a function of viewing depth. An analysis was given that relates the three solutions in terms of their accuracy and efficiency. All methods fulfill the goal of removing the aliasing artifacts, however, only (3) is efficient enough for practical application. A frequency domain analysis reveals that (3) is similar to (2). ([38])
- Formal quantitative analysis of ART and other related algebraic schemes in the new setting of low-contrast 3D reconstruction from cone-beam projection data. All implementations utilize the ray-driven splatter described above as the projection engine.

Several parameters were investigated in their impact on reconstruction quality and speed: 1. Algorithm used (ART vs. Simultaneous ART (SART)); 2. Line-integrals vs. pyramid-integrals vs. variable-size splatting kernels; 3. Value and functional variation of the relaxation coefficient λ; 4. Reconstruction grid initialization. The main results indicate that ART requires variable-sized splats to produce artifact-free reconstructions. For cone-angles less than 40° one could achieve satisfactory results without variable-size kernels by tweaking the other parameters, however, for larger cone angles variable-sized kernels are the only solution. ART is also faster, since it allows caching of previously computed results. SART, on the other hand, produces good results without the need of variable-size splats and tedious parameter tweaking, but requires 20% more computations since caching is more difficult. ([38])

- New, previously unpublished, caching schemes for ART and SART that re-utilize previously computed results. For ART, computation time can be cut into half relatively easy, without incurring much memory overhead. SART, however, requires elaborate caching schemes to keep memory requirements at a reasonable size. The savings achieved with caching and the theoretical result of [19] bring equation (1.6) near a ratio of 1.0, i.e., close the gap between ART's computational expense and that of FBP, at least theoretically. ([38])
- Development of a new projection access scheme, termed the Weighted Distance Scheme (WDS), for algebraic methods, which optimizes the projection ordering in a global sense. Quantitative comparison of WDS with all other existing projection ordering schemes is conducted. Results indicate that reconstructions generated under WDS have the least amount of noise-like artifacts and best detail fidelity. ([40])
- Accelerating ART-type methods with texture mapping hardware available on mid-range graphics workstations or PC's equipped with graphics boards. While Cabral et. al. [6] have used this kind of hardware for cone-beam reconstruction in conjunction with the non-iterative FBP algorithm, work that realizes algebraic methods on this hardware has never been published. A first implementation achieves a speedup of 10 over the optimized software implementation of [39]. By optimizing cache access behavior the speedup is improved to 75. New schemes for virtual framebuffer extensions to improve the accuracy of the projections were also described. Finally, a possible extension of the algorithm to graphics workstations with multiple processors was given. In this setting, the loading of the graphics engine and the updating of the volume data structures are handled in parallel, which should result in a further significant speedup. The texture-mapping hardware accelerated ART (TMA-ART) reconstructs a 128^3 volume from 80 projections in less than 2 minutes, with only a little decline reconstruction quality (due to the limited bit resolution of the framebuffer, which, however, may improve over the years). Hence, using this widely available form of hardware acceleration, ART is ready to be applied in real-life clinical settings, where reconstruction results are needed right away. ([45])

BIBLIOGRAPHY

[1] A. H. Andersen, “Algebraic Reconstruction in CT from Limited Views,” *IEEE Trans. Med. Img.*, vol. 8, no.1, pp. 50-55, 1989.

[2] A.H. Andersen and A.C. Kak, “Simultaneous Algebraic Reconstruction Technique (SART): a superior implementation of the ART algorithm,” *Ultrason. Img.*, vol. 6, pp. 81-94, 1984.

[3] C. Axelson, “Direct Fourier methods in 3D reconstruction from cone-beam data,” *Thesis*, Linkoping University, 1994.

[4] R.N. Bracewell, *The Fourier Transform and Its Applications*, 2nd ed., McGraw-Hill, Inc., 1986.

[5] R.N. Bracewell and A.C. Riddle, “Inversion of fan-beam scans in radio astronomy,” *Astrophysics Journal*, vol. 150, pp. 427-434, Nov. 1967.

[6] B. Cabral, N. Cam, and J. Foran, “Accelerated volume rendering and tomographic reconstruction using texture mapping hardware,” *1994 Symposium on Volume Visualization*, pp. 91-98, 1994.

[7] Y. Censor, “On variable block algebraic reconstruction techniques,” *Mathematical Methods in Tomography*, Vol. 1497, ed. G.T. Herman, A.K. Louis, and F. Natterer (Oberwolfach:Springer), pp. 133-140, 1990.

[8] R. Crawfis and N. Max, “Texture splats for 3D scalar and vector field visualization,” *Visualization’93*, pp. 261-266, 1993.

[9] C. R. Crawford and K. F. King, “Computed tomography scanning with simultaneous patient translation,” *Medical Physics*, vol. 17, pp. 967-982, 1990.

[10] M.C. van Dijke, “Iterative methods in image reconstruction,” *Ph.D. Dissertation*, Rijksuniversiteit Utrecht, The Netherlands, 1992.

[11] R. Fahrig, D. Holdsworth, S. Lownie and A.J. Fox, “Computed rotational angiography: system performance using in-vitro and in-vivo models,” *Proc. SPIE Medical Imaging 1998: Physics of Medical Imaging*, SPIE vol. 3336, 1998.

[12] L.A. Feldkamp, L.C. Davis, and J.W. Kress, “Practical cone beam algorithm,” *J. Opt. Soc. Am.*, pp. 612-619, 1984.

[13] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice.* New York: Addison-Wesley, 1990.

[14] P. Gilbert, “Iterative methods for the three-dimensional reconstruction of an object from projections,” *J. Theo. Biol.*, vol. 36, pp. 105-117, 1972.

[15] P. Grangeat, “Mathematical framework of cone-beam 3D reconstraution via the first derivative of the Radon transform,” *Proc. Mathematical Models in Tomography (Oberwohlfach, 1990), Springer Lecture Notes in Mathematics, 1497,* pp. 66-97.

[16] P. Grangeat, “Analyse d’un system d’imagerie 3D par reconstruction a partir de radiographie X en gemetrie conique,” Ph.D. disseration (Ecole Nationale Superieure des Telecommunications, Paris), 1987

[17] R. Gordon, R. Bender, and G.T. Herman, “Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography,” *J. Theoretical Biology,* vol. 29, pp. 471-481, 1970.

[18] H. Guan and R. Gordon, “A projection access order for speedy convergence of ART: a multilevel scheme for computed tomography,” *Phys. Med. Biol.*, no. 39, pp. 1005-2022, 1994.

[19] H. Guan and R. Gordon, “Computed tomography using algebraic reconstruction techniques (ARTs) with different projection access schemes: a comparison study under practical situations,” *Phys. Med. Biol.*, no. 41, pp. 1727-1743, 1996.

[20] P.E. Haeberli and K. Akeley, “The Accumulation Buffer: hardware support for high-quality rendering,” *Computer Graphics (Proc. SIGGRAPH’90)*, vol. 24, pp. 309-318, 1990.

[21] C. Hamaker and D.C. Solmon, “The angles between the null spaces of X rays,” *J. Math. Anal. Appl.*, vol. 62, pp. 1-23, 1978.

[22] K.M. Hanson and G.W. Wecksung, “Local basis-function approach to computed tomography,” *Applied Optics*, Vol. 24, No. 23, 1985.

[23] G.T. Herman, *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography.* New York: Academic Press, 1980.

[24] G.T. Herman, R.M. Lewitt, D. Odhner, and S.W. Rowland, “SNARK89-a programming system for image reconstruction from projections,” Tech. Rep. MIPG160, Dept. of Radiol., Univ. of Pennsylvania, Philadelphia, 1989.

[25] G.T. Herman and L.B. Meyer, “Algebraic reconstruction can be made computationally efficient,” *IEEE Trans. Med. Img.*, vol. 12, no. 3, pp. 600-609, 1993.

[26] G.N. Hounsfield, “A method of and apparatus for examination of a body by radiatation such as X-ray or gamma radiation,” Patent Specification 1283915, The Patent Office, 1972.

[27] W.A.Kalender, Polacin A., “Physical performance characteristics of spiral CT scanning,” *Medical Physics*, vol. 18, no. 5, pp. 910-915, 1991.

[28] A. Kaufman, Ed., *Volume Visualization*, IEEE Press 1991.

[29] S. Kaczmarz, “Angenäherte Auflösung von Systemen linearer Gleichungen,” *Bull. Int. Acad. Pol. Sci. Lett.*, A, vol. 35, pp. 335-357, 1937.

[30] A.C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988.

[31] R.M. Lewitt, “Alternatives to voxels for image representation in iterative reconstruction algorithms,” *Phys. Med. Biol*., vol. 37, no. 3, pp. 705-715, 1992.

[32] R.M. Lewitt, “Multi-dimensional digital image representations using generalized Kaiser-Bessel window functions,” *J. Opt. Sec. Am. A,* vol. 7, no.10, pp. 1834-1846, 1990.

[33] S. Matej and R.M. Lewitt, “Practical considerations for 3-D image reconstruction using spherically symmetric volume elements,” *IEEE Trans. Med. Img.*, vol. 15, no. 1, pp. 68-78, 1996.

[34] S. Matej and R.M. Lewitt, “Efficient 3D grids for image reconstruction using spherically-symmetric volume elements,” *IEEE Trans. on Nucl. Sci.*, vol. 42, no 4, pp 1361-1370, 1995.

[35] S. Matej, G.T. Herman, T.K. Narayan, S.S. Furuie, R.M. Lewitt, and P.E. Kinahan, “Evaluation of task-oriented performance of several fully 3D PET reconstruction algorithms,” *Phys. Med. Biol*, Vol. 39, pp. 355-367, 1994.

[36] E.J. Mazur and R. Gordon, “Interpolative algebraic reconstruction techniques without beam partioning for computed tomography,” *Med. & Biol. Eng. & Comput.*, vol. 33, pp. 82-86, 1995.

[37] K. Mueller, R. Yagel and J.J. Wheller, “Fast implementations of algebraic methods for the 3D reconstruction from cone-beam data,” in review, 1998.

[38] K. Mueller, R. Yagel and J.J. Wheller, “Accurate low-contrast 3D cone-beam reconstruction with algebraic methods” in review, 1998.

[39] K. Mueller, R. Yagel and J.J. Wheller, “ A fast and accurate projection algorithm for the Algebraic Reconstruction Technique (ART),” *Proceedings of the 1998 SPIE Medical Imaging Conference*, Vol. SPIE 3336, pp. , 1998. (won Honorary Mention Award.)

[40] K. Mueller, R. Yagel, and J.F. Cornhill, “The weighted distance scheme: a globally optimizing projection ordering method for the Algebraic Reconstruction Technique (ART),” *IEEE Transactions on Medical Imaging*, vol. 16, no. 2, pp. 223-230, April 1997.

[41] K. Mueller and R. Yagel, “Fast perspective volume rendering with splatting by using a ray-driven approach,” *Proceedings, Visualization’96*, pp.65-72, 1996.

[42] K. Mueller and R. Yagel, “The use of dodecahedral grids to improve the efficiency of the Algebraic Reconstruction Technique (ART),” *Annals of Biomedical Engineering, Special issue, 1996 Annual Conference of the Biomedical Engineering Society*, p. S-66, 1996.

[43] K. Mueller, R. Yagel, and J.F. Cornhill, “The weighted distance scheme: a globally optimizing projection ordering method for the Algebraic Reconstruction Technique (ART),” *IEEE Transactions on Medical Imaging*, vol. 16, no. 2, pp. 223-230, April 1997.

[44] K. Mueller, R. Yagel, and J.F. Cornhill, “Accelerating the anti-aliased Algebraic Reconstruction Technique (ART) by table-based voxel-backward projection,” *Proceedings EMBS’95 (The Annual International Conference of the IEEE Engineering in Medicine and Biology Society)*, pp. 579-580, 1995.

[45] K. Mueller and R. Yagel, “Rapid Algebraic Reconstruction Technique (ART) by Utilizing Graphics Hardware,” submitted to *IEEE Medical Imaging Conference*, November 1998.

[46] J. Neider, T. Davis, M. Woo, *OpenGL Programming Guide*, Addison-Wesley, 1993.

[47] R. Ning and S.J. Rooker, “Image intensifier-based volume angiography imaging system: work in progress,” *Proc. SPIE Medical Imaging 1993: Physics of Medical Imaging*, SPIE vol. 1896, pp. 145-155, 1993.

[48] R. Ning, Y. Zhang, D. Zhang, and D. Conover, “Image intensifier-based volume tomographic angiography imaging: animal studies,” *Proc. SPIE Medical Imaging 1998: Physics of Medical Imaging*, SPIE vol. 3336, 1998.

[49] A.V. Oppenhein and R.W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, 1989.

[50] G.N. Ramachandran and A.V. Lakshminarayanan, “Three-dimensional reconstruction from radiographs and electron micrographs: Application of convolution instead of Fourier transforms,” *Proc Nat. Acad. Sci.*, vol. 68, pp. 2236-2240, 1971.

[51] R.S. Ramakrishnan, S.K. Mullick, R.K.S. Rathore, and R. Subramanian, “Orthogonalization, Bernstein polynomials, and image restoration,” Appl. Opt, vol. 18, pp. 464-468, 1979.

[52] P. Rizo, P. Grangeat, P. Sire, P. Lemasson, and P. Melennec, “Comparison of two three-dimensional cone-beam reconstruction algorithms with circular source trajectories,” *J. Opt. Soc. Am. A*, vol. 8, no. 10, pp. 1639-1648, 1991.

[53] R.A. Robb, E.A. Hoffman, L.J. Sinak, L.D. Harris, and E.L. Ritman, “High-speed three-dimensional X-ray computed tomography: The Dynamic Spatial Reconstructor,” *Proceedings of the IEEE*, vol. 71, no. 3, pp. 308-319, 1983.

[54] R.A. Robb, " The Dynamic Spatial Reconstructor: An x-ray video- fluoroscopic CT scanner for dynamic volume imaging of moving organs," *IEEE Transactions on Medical Imaging*, vol. 1, no. 1, pp. 22-23, 1982.

[55] D. Ros, C. Falcon, I Juvells, and J. Pavia, "The influence of a relaxation parameter on SPECT iterative reconstruction algorithms," *Phys. Med. Biol.*, no. 41, pp. 925-937, 1996.

[56] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P.E. Haeberli, "Fast shadows and lighting effects using texture mapping, " *Computer Graphics (Proc. SIGGRAPH'92)*, vol. 26, pp. 249-252, 1992.

[57] O. Shochet, "Fast back-projection using hardware texture mapping with limited lit-precision," *Internal communication*, Silicon Graphics Biomedical, Jerusalem, Israel.

[58] A. Silberschatz, J.L. Peterson, P.B. Galvin, *Operating Systems*, Addison-Wesley Publishing Company, 1991.

[59] B. Smith, "Image reconstruction from cone-beam projections: necessary and sufficient conditions and reconstruction methods," *IEEE Trans. Med. Img.*, vol. 4, no. 1, pp. 14-25, 1985.

[60] D. Saint-Felix, Y. Trousset, C. Picard, C. Ponchut, R. Romeas, A. Rougee, "In vivo evaluation of a new system for 3D computerized angiography," *Phys. Med. Bio*l, Vol. 39, pp. 583-595, 1994.

[61] L.A. Shepp and B.F. Logan, "The Fourier reconstruction of a head section," *IEEE Trans. Nucl. Sci.*, vol. NS-21, pp. 21-43, 1974.

[62] J.E. Swan, K. Mueller, T. Moeller, N. Shareef, R. Crawfis, and R. Yagel, "An anti-aliasing technique for splatting," *Proceedings Visualization'97*, pp. 197-204, 1997.

[63] K. Tanabe, "Projection method for solving a singular system," *Numer. Math.*, vol. 17, pp. 203-214, 1971.

[64] H.K. Tuy, "An inversion formula for cone-beam reconstruction," *SIAM J. Appl. Math.*, vol. 43, pp. 546-552, 1983.

[65] Y. Weng, G.L. Zeng, and G.T. Gullberg, "A reconstruction algorithm for helical cone-beam SPECT," *IEEE Trans. Nucl. Sci*., vol. 40, pp. 1092-1101, 1993.

[66] L. Westover, "Interactive volume rendering," *1989 Chapel Hill Volume Visualization Workshop*, pp. 9-16, 1989.

[67] L. Westover, "Footprint evaluation for volume rendering," *Computer Graphics (SIGGRAPH)* , vol. 24, no. 4, pp. 367-376, 1990.

[68] L. Westover, "SPLATTING: A parallel, feed-forward volume rendering algorithm," *PhD Dissertation*, UNC-Chapel Hill, 1991.

[69] G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press, 1990.

[70] G.L. Zeng and G.T. Gullberg, “A cone-beam tomography algorithm for orthogonal circle-and-line orbit,” *Phys. Med.Biol.*, vol. 37, no. 3, pp. 563-577, 1992.