Stony Brook University

Efficient Reconstruction and Visualization of CT Data

A Dissertation Presented

by

Ziyi Zheng

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

December 2012

Stony Brook University

The Graduate School

Ziyi Zheng

We, the dissertation committee for the above candidate for the

Doctor of Philosophy degree, hereby recommend

acceptance of this dissertation.

Klaus Mueller, Dissertation Advisor
Professor, Computer Science Department

Xianfeng Gu, Chairperson of Defense
Associate Professor, Computer Science Department

Arie Kaufman
Distinguished Professor and Chairman, Computer Science Department

Patrick Helm
Principal Scientist, Medtronic Inc

This dissertation is accepted by the Graduate School

Charles Taber
Interim Dean of the Graduate School

Abstract of the Dissertation

Efficient Reconstruction and Visualization of CT Data

by

Ziyi Zheng

Doctor of Philosophy

in

Computer Science

Stony Brook University

2012

Cone-beam CT (Computed Tomography) has become a major imaging technique thanks to its image-fidelity and scanning time. Scientists and practitioners frequently utilize volume visualization tools for diagnosis and decision-making. The thesis work presented here seeks to improve on the volume visualization pipeline for CT generated data. We summarize our contributions into three categories.

Cone-beam CT scanners typically use analytical algorithms to reconstruct volumetric data. We studied the interpolation error of visualization tools and built a verifiable visualization tool and efficient data structure to enable users to enjoy interactive rendering speed to freely examine the high resolution data at minimal error.

For the recently developed low-dose CT which suffers from either noisy or an insufficient number of X-ray projections, we proposed an optimization framework

to determine effective parameters for the data denoising and volume reconstruction stage. We have devised an efficient method to optimize various parameters for iterative CT reconstruction using an ant colony optimization algorithm. We also developed an interactive user interface to visually explore various acquisition settings. Our preliminary results show that the learned parameters can be readily applied to similar scans with promising results.

Lastly, we provide visual guidance which can boost user efficiency when exploring the data. For guided visualization, we propose a view suggestion framework rooted in high-dimensional feature space which does not rely on particular transfer functions or volume segmentations as an initial input.

To My Parents Xiangyang Zheng and Suyu Yang

My Grandparents and My love Luxi Ji

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to sincerely thank my thesis advisor, Professor Klaus Mueller for his guidance on research. He led me to the field of scientific visualization and GPU computing. I cannot imagine myself completing this dissertation without his inspiration and encouragement.

I would like to thank Professor Arie Kaufman and Professor Xianfeng Gu for their invaluable inspirations and discussions which I have substantially benefited from during my Ph.D. studies.

I would like to thank Patrick Helm, Shuanghe Shi, Josh Star-Lack and Minshan Sun for their support in my research. Working with them was such a pleasure and I gained incomparable hand-on experience. And I would like to thank Patrick Helm especially for taking his time to serve as the external member of my dissertation committee.

I would also like to thank my colleagues in computer science department especially those in center of visual computing, Eric Papenhausen, Wei Xu, Yun Zeng, Tingbo Hou, Ruirui Jiang, Nafees Ahmed, Sungsoo Ha, Zhiyuan Zhang, Supriya Garg, Kaloian Petkov, Joeseph Marino, Krishna Chaitanya Gurijala, Hongyu Wang, Xiaotian Yin, Feng Qiu, Zhe Fan, Wei Zeng, Min Zhang, Xiang Cai, Jun Yuan, Xiaomeng Ban, Rui Shi, Xin Zhao, Bo Li, Lei Wang and Tianyun Ling for delightful collaborations and discussions we had together.

Last but not least, I want to thank my parents, grandparents and my wife. Without their support, this thesis would not have been possible.

List of Publications

Journal Papers

1. Nafees Ahmend, **Ziyi Zheng**, and Klaus Mueller, "Human computation in visualization: using purpose driven games for robust evaluation of visualization algorithms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2104-2113, 2012.
2. **Ziyi Zheng**, Nafees Ahmed, and Klaus Mueller, "iView: a feature clustering framework for suggesting informative views in volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1959-1968, 2011.
3. **Ziyi Zheng**, Wei Xu, and Klaus Mueller, "VDVR: verifiable volume visualization of projection-based data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1515-1524, 2010.
4. Chunhua Men, Xuejun Gu, Dongju Choi, Amitava Majumdar, **Ziyi Zheng**, Klaus Mueller, and Steve B. Jiang, "GPU-based ultra fast IMRT plan optimization," *Physics in Medicine and Biology*, vol. 54, no. 21, pp. 6565-6573, 2009.
5. Eric Papenhausen, **Ziyi Zheng**, and Klaus Mueller, "Creating optimal code for GPU-accelerated CT reconstruction using ant colony optimization," *Medical Physics* (accepted in 2012).

Conference Paper

6. **Ziyi Zheng**, Eric Papenhausen, and Klaus Mueller, "Searching effective parameters for low-dose CT reconstruction by ant colony optimization," in *Proceedings of the second CT meeting*, pp. 182-185, 2012.
7. Eric Papenhausen, **Ziyi Zheng**, and Klaus Mueller, "Coding ants: using ant colony optimization to accelerate CT reconstruction," in *Proceedings of the second CT meeting*, pp. 356-359, 2012.

8. **Ziyi Zheng** and Klaus Mueller, “Identifying sets of favorable projections for few-view low-dose cone-beam CT scanning,” in *Proceedings of The 11th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, pp. 314-317, 2011.
9. **Ziyi Zheng**, Wei Xu, and Klaus Mueller, “performance tuning for CUDA-accelerated neighborhood denoising filters,” in *Proceedings of the Workshop on High-Performance Image Reconstruction*, pp. 52-55, 2011.
10. Eric Papenhausen, **Ziyi Zheng**, and Klaus Mueller, “GPU-accelerated back-projection revisited: squeezing performance by careful tuning,” (**Best Paper**) in *Proceedings of the Workshop on High-Performance Image Reconstruction*, pp. 19-22, 2011.
11. **Ziyi Zheng**, John Pavkovich, Mingshan Sun, and Josh Star-Lack, “Fast 4D cone-beam CT reconstruction using the McKinnon-Bates algorithm with truncation correction and nonlinear filtering,” (**Honorable Mention Poster**) in *Proceedings of SPIE*, vol. 7961, pp. 79612U, 2011.
12. **Ziyi Zheng** and Klaus Mueller, “Reconstruction and visualization of model-based volume representations,” in *Proceedings of SPIE*, vol. 7625. pp. 76251O, 2010.

Chapter 1. Introduction

1.1 Problem Statement

Computed Tomography (CT) is a widely employed imaging modality in medical and industrial application. Many researchers and practitioners make evaluations or diagnosis decisions by examining CT scanned data. For many of these applications, high accuracy and fast speed are of major importance.

Visualization researchers and practitioners either know or at least suspect that their visualization tools may not be completely truthful to the underlying data. This is often rooted in compromises that need to be made for balancing rendering speed and quality. An often-studied error source is the interpolation of off-grid samples, where aliasing can lead to misleading artifacts and blurring, potentially hiding fine details of critical importance. As for CT, the volumetric data subject to interpolation in the rendering stage are often not the actual raw data, but only derived from them. This implies the quality-enhancing effort in visualization can be rooted on interpolation in X-ray projections. However, traditionally volume rendering algorithms favor grid-only data in terms of computational cost. There is a crucial need of an efficient rendering method taking consideration of projection domain data.

Before visualization stage, CT reconstruction as a data-generation that takes X-ray projections and create 3D volumetric data with clinical/industrial value. With the growing concerns of the X-ray radiation to the patients in medical applications, more and more researches have been interested in lowering the total number of projection or reducing the radiation dose per projection. To cope with the slower reconstruction speed and lower quality, computational-efficient regularization algorithm becomes a hot topic. There are a great amount of image denoising algorithms. But most of them require manually chosen parameters thus cannot be claimed as general solutions. Automatically choosing effective parameters to perform image processing remains a challenging problem in data generation stage.

During the visualization stage, the visual exploration and extraction of relevant information from 3D volume data can be a daunting task as it often requires users to try out many different combinations of views and transfer functions. In this regard, having proper views to start with can greatly improve the efficiency of the data exploration process. Hence, given an arbitrary volume dataset, the suggestion of a set of interesting views has been a research topic of great interest, but also one of challenges.

1.2 Approach

To address the three problems above, we propose corresponding improvements in data generation domain, data transformation and visualization domain. These algorithms can be illustrated in a general framework listed below.

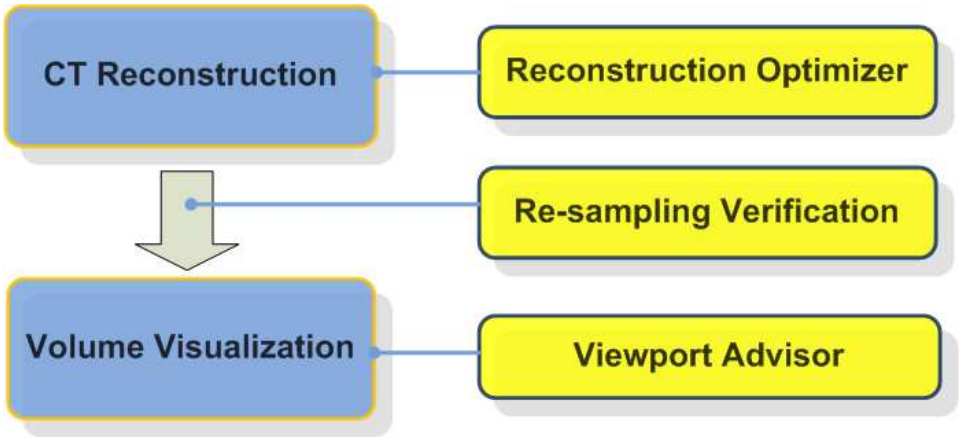


Figure 1. The overview of the proposed reconstruction and visualization framework.

As seen in Figure 1, the CT reconstruction optimizer focuses on an automatic parameter tuning. In the pre-computation step of our parameter learning framework, effective parameters are learned with the access of a known gold-standard. As these domain-specific and algorithm-dependent parameters are obtained, they are applied in the similar incoming data. In addition, an edge-enhancement component is introduced and automatically tuned to increase the sharpness in iterative reconstruction. The implementation details of the automatic parameter optimization method can be found in Chapter 3. Chapter 2 presents the concept and implementations in CT reconstruction. It

also includes discussions about acceleration techniques used in GPU-based cone beam CT reconstruction.

Aiming to account for visualization errors directly in the volume generation stage, we propose a verifiable visualization method. It informs the CT reconstruction process of the specific filter intended for interpolation in the subsequent visualization process, and this in turn ensures an accurate interpolation there at a set tolerance. Chapter 4 discusses the proposed verifiable visualization in details.

Finally, the automatic viewpoint advisor — iView — suggests promising views based on the high-dimensional clustering. It informs users of promising views before laborious transfer function exploration even begins and so prevents “dead-end” transfer function exploration experiences. Our view suggestions are adaptive to what the user has already seen. Users can explore the entire view-space with progressive suggestions of promising viewpoints. This facilitates a fully unconstrained volume exploration, but ensures that all important features are eventually seen. In addition the viewpoint advisor provides viewpoint set solutions that are optimal. Chapter 5 presents this view suggestion method.

Finally in Chapter 6, we summarize the contribution and provide potential directions for future work.

Chapter 2. Computed Tomography

2.1 Analytical Reconstruction

Filtered back-projection (FBP) is a popular analytical reconstruction method in CT. The theoretical bases of FBP are Radon transform and Fourier slice theorem. Here we use a 2D case for ease of illustration (pictured Figure 2). Any point within the circle defined by the intersection of all detector shadows can be accounted for in the X-ray projections and later be reconstructed via FBP. The back-projection is essentially an inverse (X-ray) volume rendering, that is, a volume point is calculated by summing the contributions of all rays that emanate from the corresponding projection pixels and pass through the point. For parallel-beam projection geometries, the Fourier transform for each projection constitutes a radial slice of the imaged signal's Fourier spectrum, as shown Figure 2(b) for the amplitude portion. This spectrum is available on a polar grid and is bandlimited to $\pm N/2$, where N is the projection's resolution (the number of pixels). This existence of a bandlimit will play an important role in error analysis in Chapter 4. Also note that the outer circle areas are less tightly sampled. This can be compensated for by pre-filtering the data by a ramp (Ram-Lak) filter, or more noise-suppressing filters, such as the Shepp-Logan [80], which multiply the ramp by a window to de-emphasize high frequencies.

The analytical 2D filtered back-projection formula is:

$$f(x,y) = \int_0^{\pi} \int_{-\infty}^{+\infty} P_\theta(\omega) |\omega| e^{j2\pi\omega t} d\omega d\theta = \int_0^{\pi} q_\theta(t) d\theta \qquad (1)$$

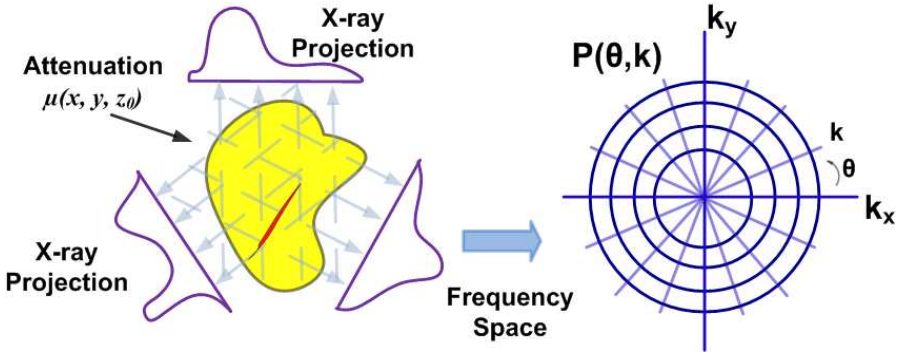
where $t = x\cos\theta + y\sin\theta$ is the projected location of point (x, y) and θ is the projection angle. $P_\theta(\omega)$ is the 1D Fourier transform of a 1D projection $p_\theta(t)$. The inner integral is the 1D inverse Fourier transform resulting in $q_\theta(t)$. The outer integral represents the summed back-projection of all $q_\theta(t)$.

In practice,

$$f(x,y) \approx \frac{\pi}{K} \sum_{i=1}^{K} q_{\theta_i}(x\cos\theta_i + y\sin\theta_i) \qquad (2)$$

where $q_{\theta_i}(t)$ are the ramp-filtered projections. Equation (1) and Equation (2) formalize the reconstruction pipeline. For each 1D projection $p_{\theta_i}(t)$, we first apply a 1D Fourier transform to yield $P_{\theta_i}(\omega)$. Each $P_{\theta_i}(\omega)$ is ramp filtered and then an inverse Fourier transform computes $q_{\theta_i}(t)$ (for 2D data the ramp filtering needs to be done along the projection rows in 1D). Finally, we sum all back-projected $q_{\theta_i}(t)$ at the voxels and divide this sum by the number of projections K. This K should be at least $(\pi/2)N$ such that the polar Fourier transform has about the same resolution in θ at the periphery and K (see Figure 2(b)) [80]. Filtered back-projection algorithms can be generalized in Cone-beam geometry. Feldkamp-Davis-Kress (FDK) algorithm [29] is one popular method.

The filtered backprojection algorithm is well adapted to GPU. The RabbitCT benchmark [71] shows the speed of 512^3 volume for a given projection data using FDK. The current fastest solution is GPU in around 5 seconds [65].



(a) (b)

Figure 2. The X-ray transform and its Fourier spectrum in 2D.

2.2 Iterative Reconstruction

The filtered-backprojection algorithms can provide high resolution results but requires several hundreds of patient X-ray projections. With the growing concern about the potential risk of X-ray radiation exposure to the human body, low-dose CT has become a significant research topic. Dose reduction usually involves lowering the X-ray energy per projection and/or reducing the total number of projections. Both methods typically suffer from low signal-to-noise ratio (SNR) in the reconstructions. Iterative

reconstruction schemes [95] [60] have been shown to cope well with these few-view or high-noise scenarios.

Iterative reconstruction consists with two basic components: forward projection and back-projection. The forward projection operator simulates X-ray images at a certain viewing angle θ. The result of this projection is then compared to the acquired image obtained at the same viewing configuration. Here, the weight factor w_{il} determines the contribution of a voxel v_l to a ray r_i is given by the interpolation kernel used for sampling the volume. The forward projection can be formulated as:

$$r_i = \sum_{l=1}^{N} w_{il} \cdot v_l \qquad i = 1,2,\ldots,M \tag{3}$$

where M and N are the number of rays and voxels, respectively. For OS-SIRT [95] algorithm, the correction update is computed as:

$$v_j^{(k+1)} = v_j^{(k)} + \lambda \frac{\sum_{p_i \in OS_s} \frac{p_i - r_i}{\sum_{l=1}^{N} w_{il}} w_{ij}}{\sum_{i=1}^{N} w_{ij}} \tag{4}$$

$$r_i = \sum_{l=1}^{N} w_{il} \cdot v_l^{(k)} \tag{5}$$

Here, p_i represents the pixels in the M/S acquired images that form a specific (ordered) subset OS_S where $1 \leq s \leq S$ and S is the number of subsets. The factor λ is the relaxation factor that scales the corrective update to each voxel. This factor is important in balancing quality and speed and will be optimized by our parameter tuning in Chapter 3. The factor k is the iteration count, where k is incremented each time after all M projections have been processed.

The iterative reconstruction algorithm is computational intensive task. It is beneficial to use GPU to accelerate the forward projection. The forward projection as X-ray projection simulator uses a ray-driven approach, where each ray is mapped to a parallel thread and interpolates the voxels on its path. However, the back-projection still uses a voxel-driven approach, where each voxel is mapped to a parallel thread and interpolates the 2D correction projections. Thus, the weights used in the projection and the back-

projection are slightly different but it has minimum effect on the reconstruction quality [97] [101].

2.3 Regularization

One of challenges of low dose CT reconstruction is the lack of sufficient data. The regularization algorithm usually employed is Total Variation Minimization (TVM) as used in the Adaptive-Steepest-Descent-Projection-Onto-Convex-Sets (ASD-POCS) algorithm[76]. Besides TVM, de-noising filters such as the bilateral filter (BF) [84] and the non-local means (NLM) filter [9], can also be used in regularization [97] [98] [109]. Here we discuss these two image denoising filters. The bilateral filter (BF) [7] is an edge-preserving non-linear filter. It replaces a pixel's value to the weighted average of nearby pixels. The weighting is a multidimensional Gaussian considers both the pixel-location's distance and pixel-value's difference. A fixed window area is usually used to achieve fast and accurate computation:

$$BF(x) = \frac{\sum_{t \in W} c(t)s(f(x), f(x+t)) \cdot f(x+t)}{\sum_{t \in W} c(t)s(f(x), f(x+t))} \tag{6}$$

$$c(t) = \exp(-\frac{\|t\|^2}{2 \cdot \sigma_d^2}) \tag{7}$$

$$s(f(x), f(x+t)) = \exp(-\frac{(f(x) - f(x+t))^2}{2 \cdot \sigma_r^2}) \tag{8}$$

Here, t and x represent the spatial variables. W is the window centered at x. f is the input image. The multi-dimensional Gaussian weighting function can be separated into two parts: the measured closeness and pixel value similarity. The closeness function c is a spatial filter to average nearby pixels while the similarity function s is a range filter used to exclude dissimilar pixels. σ_d and σ_r control the amount of smoothing. Normalization forces the sum of pixel weights to 1.

The NLM filter is another non-linear filter. Intuitively, it replaces the pixel located at x with the mean of the pixels whose Gaussian-weighted neighborhood looks similar to the neighborhood of x:

$$NLM(x) = \frac{\sum_{y \in W} \exp(w(x,y)) \cdot f(y)}{\sum_{y \in W} \exp(w(x,y))} \tag{9}$$

$$w(x,y) = -\frac{\sum_{t \in N} G_a(t) | f(x+t) - f(y+t) |^2}{h^2} \tag{10}$$

Here, x, y and t are spatial variables. W is the window centered at x. N is the neighborhood centered at x or y. G_a is a Gaussian kernel of zero-mean and a standard deviation a. The variable h acts as a filtering parameter and when it is increased, the weights to dissimilar pixels are increased to allow for more smoothing. Thus, the NLM filter contains several parameters to achieve best quality.

In contrast to the BF, the NLM removes the spatial smoothing form but increases the dimension of the range filter. The comparison of the similarity of two neighborhoods, represented by the high dimensional vectors is applied. This modification brings more accuracy to the de-noising but costs much more time to compute.

2.4 GPU Accelerations

2.4.1 SIMT Architecture

Here we take the NVIDIA GeForce GTX 480 GPU as an example to discuss the GPU architecture. A GTX 480 GPU card contains 480 processors. These 480 processors are grouped into 15 *streaming multi-processor* (SMP) which can perform tasks independently from each other. As shown in Figure 3, each SMP contains 32 processors, which allow 32 threads (a warp) to execute concurrently. Thus each SMP is inherently based on single instruction multiple threads (SIMT) design. In the best case, the GTX 480 has theoretical computational power reaching 1.3 Tera-Floating Point Operations Per Second (TFLOPS) in single floating-point precision which largely outperform the CPU computational power.

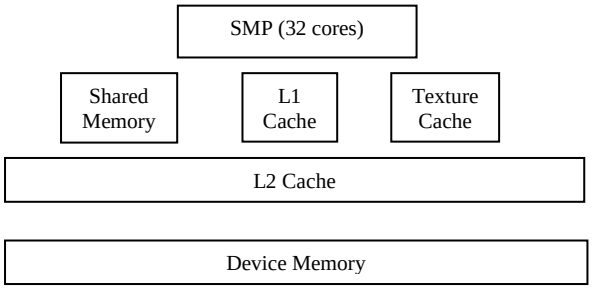


Figure 3. The GPU processor-memory configuration for one SMP

GPU device memory is an off-chip memory that stores the input data and receives the output from the processors. The GTX 480 has 1.5GB DDR5 device memory with peak bandwidth 177.4 GB/s. Although the bandwidth of GPU memory is much faster than that of the CPU memory, it has several limitations. First, each off-chip memory (also called device/global memory) access instruction takes several hundreds of clock cycle. This latency needs to be alleviated by issuing a large amount of threads which will automatically enable hardware context switching. Second, the memory instructions should better to be coalesced or at least have a specified granularity (128 bytes). The maximum GPU global bandwidth can only be achieved by issuing 1 memory instruction for 128 bytes data. This implies 32 neighbouring threads (a warp) should read/write within a 128-byte-aligned segment. With proper alignment, sequential mapping of threads to memory address will yield a coalesced memory access pattern.

To further reduce the huge costs associated with off-chip memory access, the cache can be leveraged. Constant memory cache is the simplest type of cache. It is an off-chip memory with the similar bandwidth as device memory. To speed up the constant data access rate, a GTX 480 contains an 8KB cache per 8 processors for constant memory access. Besides the constant cache, 32 processors within one SMP share an L1 cache and a user-controllable cache known as the shared memory as shown in Figure 3. The difference between the L1 cache and the shared memory is that the former is automatically scheduled by the hardware and the latter can be controlled by the user to perform prefetching. The amount of shared memory and L1 cache in one SMP is user-configurable (16KB + 48KB or 48KB + 16 KB).

NVIDIA GPUs can be programmed via a C-like API —CUDA. CUDA is a general purpose API which exposes more control over how a task is computed on the GPU hardware, as compared to graphics-based APIs (CG, GLSL). The task-hardware mapping is enabled by introducing the concept of "*block*". Each *block* is mapped to an SMP.

The key difference between CPU implementation and its GPU counterpart is the parallel programing. While typically CPU program will launch one thread, GPU will launch millions of threads with the same instruction. A large amount of threads are executed in terms of thread *blocks*, whereas the total task is called *grid*. On the hardware level, each *block* is mapped to a single SMP. In the back-projection stage of the CT reconstruction, SMPs are assigned to different regions of the resulting volume sequentially. This enables a mapping where the *grid-block* decomposition in CUDA corresponds to the volumetric reconstructed 3D dataset. To avoid misunderstanding, we use *block* and *grid* only as terms in CUDA, not for their geometry meaning.

2.4.2 GPU Accelerated Back-Projection

Recent researches focus on how to perform the computation on parallel computing devices, to yield the speed satisfying the clinical need. GPU implementation is a feasible high-performance solution along with other choice, such as Cell processors and FPGAs. For the analytical reconstruction algorithm, such as FDK, GPU-based solutions can match the speed of data generation by X-ray scanners [33] [94]. Nevertheless, for low-dose CT which typically requires iterative reconstructions, GPU-based reconstructions have a much longer running time [41]. In the back-projection computation, the commonly used method is the voxel-driven method. For example, slabs in the volume (Figure 4) are mapped to thread *blocks* in CUDA. This volume-to-slabs decomposition has been used in [33] [41] and [74]. Another type of mapping is based on decomposing the volume into horizontal tiles and each tile is mapped to a CUDA thread *block*, as used in [63] and [64]. While the former use 2D square tiles, the latter uses 1D linear tiles. We take the slab-based approach as an example to show how to optimize the hardware mapping to improve cache hit rate.

Each 3D slab of the reconstructed-volume is assigned to a SMP. In the depth

dimension, the brick extends through the reconstructed volume. The slab is further divided into vertical tiles. Here, the product of the tile's width *w* and height *h* needs to be below the maximum number of threads per *block L*. In the NVIDIA Fermi GPU, *L* = 1024. The tile's width should be a multiple of 32 to ensure a coalesced writing pattern.

When performing the back-projection inside each slab, the order of execution of this back-projection follows along the set of 2D vertical tiles arranged in depth order. After we finish back-projecting, we incrementally store the resulting slices back in global memory.

The mapping of the kernel to the projection geometry is depicted in Figure 4. The concurrent execution of threads inside an SMP is limited by the maximum number of threads that can be run there. The NVIDIA 480 has 15 SMPs. A set of projections i_a to i_b is processed at the same time. Then the total projected area processed by the GPU, arising from projection i_a to projection i_b is:

$$\sum_{i=i_a}^{i_b} A_i = 15l^2\left(\sum_{i=i_a}^{i_b} MT|\cos(\theta_i)|\right) \tag{11}$$

In Equation (11), MT is the maximum resident threads for each SMP, θ_i is the projection angle for i^{th} projection and l is the ratio of source-object distance over source-detector-distance. Assume C_{cache} is the constant representing the amount of L2 caches, the projected area should be smaller than the cache limit (Equation (12)) to ensure better cache hit-rate.

$$\sum_{i=i_a}^{i_b} A_i \le C_{cache} \tag{12}$$

Many back-projection implementations ([33] [41] [63] [64] [74] [94]) use texture memory for projection data storage to deal with irregular memory fetching pattern. Using texture memory has several advantages. First, the trilinear/bilinear interpolation is supported at hardware level, which can reduce the computation cost of GPU processors. Second, hardware-scheduled cache can benefit data reading with locality. Last, except locality, texture memory has no other constraint such as coalescing or confliction.

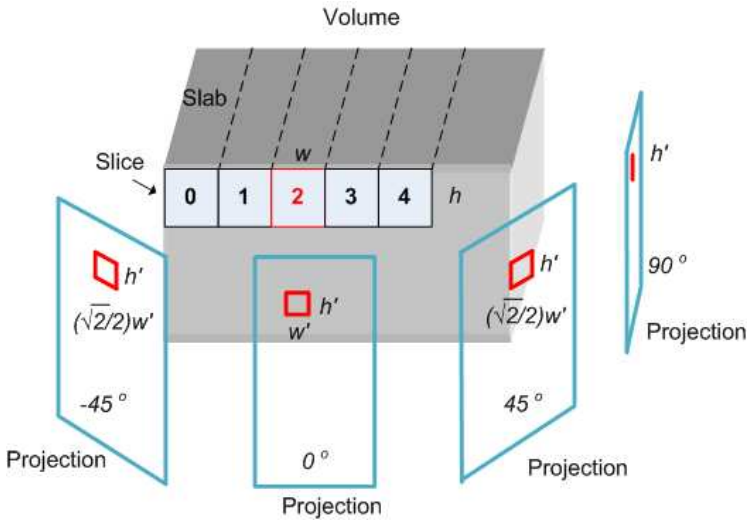


Figure 4. Illustration of bricks, slices, slices' projections and corresponding SMPs. In the volume, different numbers represent the different SMPs. The red color indicates the input and output regions involved in SMP number 2.

2.4.2.1 Back-Projection Ordering

According to different orderings of back-projection loops, the published methods can be classified into three categories: single projection method, multiple projections method and hybrid ordering method.

```
FOR each projection
  FOR each (slab) slice along the depth dim
    Accumulate projected values in the current slice
    Write results to current slice
  END
END
```

Figure 5. The single projection method.

Figure 5 shows the single projection method. The idea behind this scheme is to obtain a maximum reuse of the SMPs' L2 cache holding the projection. The locality of the texture reading is increased since the amount of overspreading of the reading location is restrained. Then for each slice there is a cache-miss at the beginning but not likely thereafter. The disadvantage of this method is memory-writing overhead. N projections will require reading and writing the computed results N times. Also, when we perform

incremental writing on the global memory, the L2 cache will also cache the output slices. This undesirable cache behavior will reduce the effective capacity of L2 cache. Keck et. al. [41] used this method in CUDA-based SART. This method is better when cache-miss penalty is larger than memory writing overhead.

```
FOR each (slab) slice
  FOR each projection
    Accumulate projected values in the current slice
  END
  Write results to current slice
END
```

Figure 6. The multiple projections method.

Figure 6 shows the multiple projections method. Xu and Muller [94] used this method but their work was based on the classic graphics pipeline. This method does not have writing output overhead (N projection only need to write the computed result once). The cache performance will be the bottleneck of this approach. The downside of this method is that cache for the input projection will be depleted very fast. Noël et. al. [63] also used this method in their tile-based decomposition CUDA kernel. This method is better when cache-miss pentaly is less than memory writing overhead.

```
FOR each set of N/J projections
  FOR each (slab) slice
    FOR each J projections
      Accumulate projected values in the current slice
    END
    Accumulate results to current slice
  END
END
```

Figure 7. The hybrid ordering method.

Figure 7 shows the hybrid ordering method. Okitsu et. al. [64] used this method in their tile-based decomposition kernel. This method requires the trade-off between better cache and less output. The hybrid ordering improves the cache hit rate. The downside of this method is that the innermost loop for *J* projections is usually unrolled to avoid conditional branches. But it will require more registers and will result in register spilling which will reduce concurrency in the GPU.

This method helps explore the balance between single-projection and multiple-projections. The innermost loop number J needs to be small enough to not exceed the L2 cache size (Equation (12)). On the other hand, J needs to be large enough such that the number of slice updates in global memory is minimized.

2.4.2.2 Cache Optimization

As for projection angles, there can be two cases:

1. Back-Projection within [45,135), [225, 315) degrees. A 2D slice with area $w \times h$ is projected into a region with area within $[0, (\sqrt{2}/2)w' \times (\sqrt{2}/2)h']$, the cache problem is not very severe. Problem only arise when projections are concentrated on 45+90k degrees, where k ∈ N.
2. Back-Projection around [135,225), [-45, 45) degree. A 2D slice with area $w \times h$ is projected into a region with area within $[(\sqrt{2}/2)w' \times (\sqrt{2}/2)h', w' \times h']$. Then the SMP will be depleted of cache frequently.

We propose a method using a 3D transpose to improve the cache-hit rate. Note that rotating the volume by 90 degree will change case 2 into case 1, but writing the results directly into global memory will cause the coalescing problem. The reason can be explained for the 2D case since we essentially switch the row-major storage into column-major, which is not sequentially stored in memory any more but scattered into different memory segments. We use shared memory as a buffer to perform the matrix transpose inside each SMP before writing to the device memory, to preserve the coalescing pattern of the output. Also, we avoid shared memory bank conflicts by using a 33×4 byte pitch. This will facilitate 32 parallel memory-conflict free threads. Another copy of the volume need to be allocated in the device since the 3D transpose is not in-place.

The reshuffling from XYZ to ZYX is listed below:

1. Load a 2D XZ slices into shared memory
2. 2D transpose to ZX slices
3. Write to ZYX in device memory

The implementation was based on NVIDIA's CUDA 3.2 and on the GTX 480. All CUDA Kernels had the same configuration: 32×8 threads per block. The inputs were 364 1024×768 projections on a half circal trajectory. We reconstructed a 3D volume within the FOV with two different resolution, 256^3 and 512^3. We tested the different running times for the Single projection method (S), Multiple projection method (M) and Hybrid ordering method (H). We then added the comparison to our optimized 3D transpose methods (T). As shown in Table 1, our transpose method had better performance regardless of the back-projection loop-ordering.

Table 1. The running time for 364 projections (in millisecond)

Data Size	S	M	H	S+T	M+T	H+T
256^3	1110	1018	1181	785	688	785
512^3	4814	4880	5001	4360	4263	4060

The 3D Transpose method used 16x16 CUDA blocks. Although it wasted some bandwidth, as opposed to the 32x32 CUDA block, since the memory accesses were not fully aligned, the 16x16 configuration was experimentally faster than 32x32 and it achieves ocupancy 1. Our 3D transpose method took 2ms for a 256 volume and 20ms for a 512^3 Volume.

Table 2. The final results and speedup

Data Size	Our (in millisecond)	Projection/sec	Speedup
256^3	688	529	1.48
512^3	4060	90	1.18

The final back-projection performance is shown in Table 2. For a 256^3 volume, our remapping method with shared-memory-enabled transpose achieved a total speedup of 1.48. For a 512^3 volume, our remapping method achieved a total speedup of 1.18.

Besides NVIDIA, other major GPU manufactures are ATI and Intel. Here we focused on NVIDIA's GPUs to evaluate different acceleration techniques. Although the specification of GPUs may change, the concept and algorithm can generally port to a different GPUs by using a cross-vendor programming API – OPENCL. The volume reshuffling scheme can improve the locality of the memory reading pattern in GPU-based back-projection. The optimized scheme can yield better cache hit rate and can be added to variety of existing methods with different back-projection ordering ([33] [41] [63] [74]), except for those implementations that use a square-tile in block-level mapping [64]. The results show our method is particularly useful for reconstructions with low-resolution volume with high-resolution projections.

2.4.3 GPU Accelerated Regularization

Neighborhood filter CUDA kernels are similar to their CG implementations — fragment programs. If we assume one CUDA kernel function only computes one resulting pixel, a neighborhood filter fragment program can be changed into its CUDA kernel without much modification. In the SIMD architecture, the same kernel/fragment program will replicate itself to all different processors. These threads on different processors have unique two-dimensional IDs (x, y) to guide them to read neighborhood data around (x, y) and output to the value at (x, y).

Here we list the pseudo-code for a 2D neighborhood filter kernel:

```
Neighborhood_filter_2D
Obtain the current thread ID (x, y)
          Collect all pixels' values in 2D neighborhood within the mask
Calculate output pixels value defined by filtering algorithm
Output results (x,y) at the resulting image
End
```

Figure 8. Pseudo-code for 2D neighborhood filter kernel.

CUDA has more sophisticated controls which are not available in CG. CUDA's execution configuration guides how the parallel computations are assigned on GPU

hardware on streaming-multi-processor (SMP) level. This can be done by dividing the 2D image into tiles and assign them to a CUDA *block*. Each of the 2D tiles will be mapped into a SMP.

To achieve maximum bandwidth in reading, the output image is stored in 2D pitched memory and the input is stored in a read-only 2D texture. In addition, to confirm the rule that each warp (32 threads) writes to a 128-byte segment, each thread should output a 4-byte unit. This 4-byte unit can be 4 characters, 2 short integers or 1 single-precision floating-point number.

2.4.2.3 Pre-computation

Some of neighborhood filters such as the bilateral filter or the non-local means (NLM) filter involve 2 Gaussian weights: σ_x, σ_y. They define the smoothing parameters in the x, y axis respectively.

Pre-computing techniques can be applied on the filter to reduce computational cost. Given the mask size, we can pre-compute a discrete mask for the 2D Gaussian smooth kernel and store it in the GPU's constant memory. Then once cached in SMP, these pre-computed weights will be ready to use which will save a huge amount of exponential computations. However, the Gaussian in the intensity domain which is inherently different from spatial dimension since it is sampled in a continuous domain. Although similar pre-computing method exists, which discretize the continuous intensity domain and lookup the pre-computed weightings, we have not explored the speed-quality trade-off of this approximation technique. We calculate intensity Gaussian on the fly, therefore let our GPU algorithm is an exact method.

We store the output volume in 2D pitched memory in order to achieve better global memory bandwidth. The output is decoupled from the order of the loops in the CUDA kernel computation. Switching the order of the loops or changing the output storage to YX will result in non-coalesced memory writing patterns that downgrade the performance. Furthermore, this loop order also indicates the pre-computed weights should be organized in XY order.

2.4.2.4 Prefetching

We also use the prefetching method to reduce the data-transfer cost, based on huge difference between on-chip and off-chip memory bandwidth. Prefetching is done according to the *apron* which is the image region served as input of a *block* of threads. The size of the 2D preloading apron is:

$$(w_b+2r_w+2r_p)\cdot(h_b+2r_w+2r_p) \tag{13}$$

where w_b,and h_b are width and height of a 2D CUDA block. r_p is the patch radius and r_w is the windows radius in non-local mean filler [9], while for bilateral filter [84] $r_p = 0$.

The apron is usually larger than the output region and thus aprons from different CUDA *blocks* are overlapped. Since neighborhood filters re-use input data in an apron multiple times, shared memory can serve as a user controllable cache to reduce the off-chip memory access. Then a 2D prefetching approach can yield less cache misses which will result in better performance.

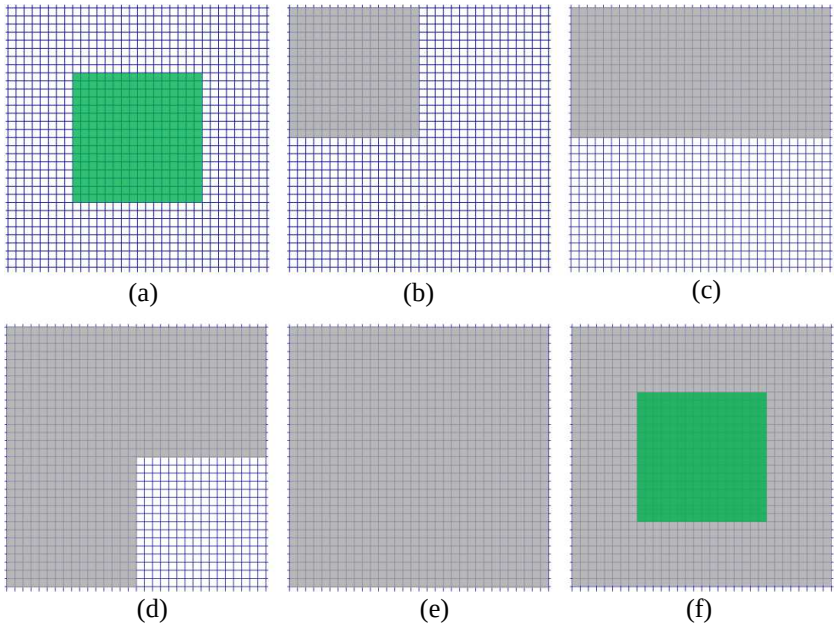


Figure 9. An example to illustrate prefetching procedure. (a) shows the configuration with one CUDA block (in green). (b-e) show the neighborhood pixels are loaded into shared memory (in gray). (f) shows the data configuration after loading all the input into the user shared memory.

Figure 9 illustrates the prefetching scheme for the 2D neighborhood filters. In this case, a 16×16 CUDA block (in green) needs to read 32×32 pixels in its neighborhood (all pixels in panel (a)). Performing neighborhood filtering directly on (a) will result in low performance. Panel (b-e) shows the proposed prefetching method will load 4 16×16 tiles into shared memory (in gray) in sequence. Finally in panel (f), the CUDA threads in the green CUDA block can fast access the input in on-chip cache (shared memory). Then applying neighborhood filters on (f) will guarantee there will be no cache miss afterward thus will boost the performance.

2.4.2.5 Experiments

Our experiments were conducted on an NVIDIA GTX 480 GPU, programmed with CUDA 3.2 runtime API and with an Intel Core 2 Duo CPU @ 2.66GHz. We built the program in 32bit mode. In the experiment, the size of the CUDA *block* is set to 32×32. The first dimension is chosen to be 32 to conform to the coalescing rule. The second dimension we choose the maximum number as 32 due to the block's size limit 1024 in the NVIDIA Fermi card.

We did a performance and image quality study on one slice of a human head. We simulated 90 parallel beam projections and added Gaussian noise SNR=25 (SNR is computed by the ratio of the mean pixel value to the standard deviation of Gaussian noise) into the projections. Figure 4(a) shows the gold-standard and Figure 4(b) shows the iterative reconstruction results from noisy projections.

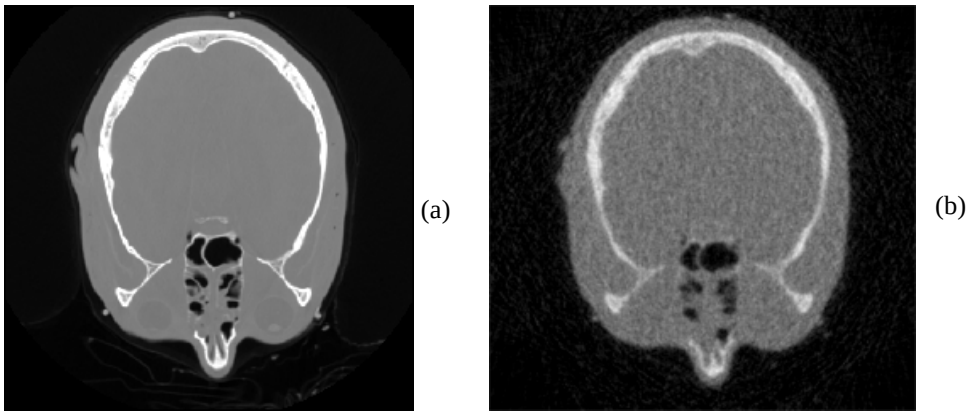
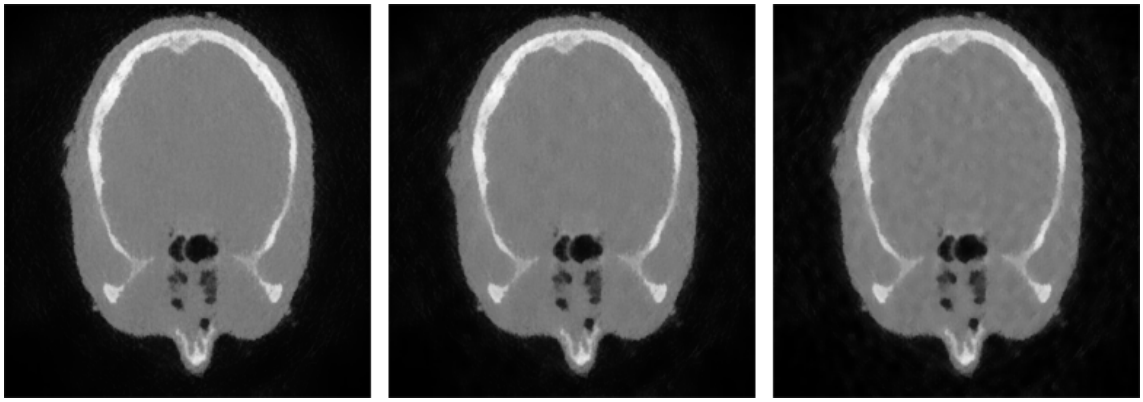


Figure 10. Testing image with size 256^2. (a) shows the gold-standard. (b) shows iterative reconstruction from 90 noisy projections.

Figure 10 shows images restored bilateral filtering. The image quality of the bilateral filtering depends on smoothing parameters and window sizes. Here σ_x and σ_y control the spatial Gaussian, σ_r controls the range Gaussian and r_w is the window radius. The window size is $2r_w+1$. Here we show approximately the best parameters for each window size. The large window size case (17×17 in Figure 11(a))) generated better results than 11×11 (Figure 11(b)) and 7×7 (Figure 11(c)).



(a) $\sigma_x = \sigma_y = 30$ $\sigma_r = 19$ $r_w = 8$

(b) $\sigma_x = \sigma_y = 38$ $\sigma_r = 19$ $r_w = 5$

(c) $\sigma_x = \sigma_y = 40$ $\sigma_r = 20$ $r_w = 3$

Figure 11. Bilateral filtering results.

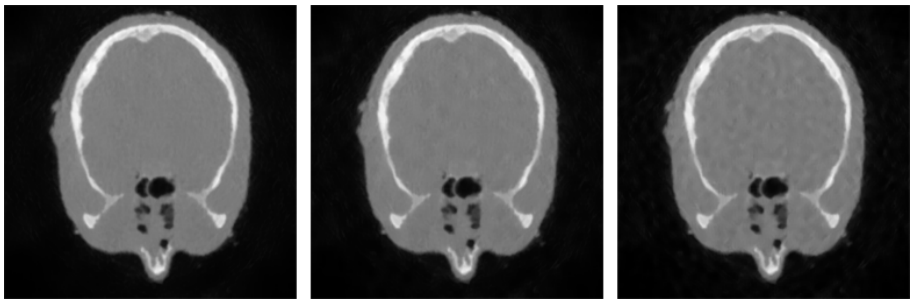
We extend the performance test of bilateral filter on larger image sizes. The computation time is listed in Table 3(in millisecond). Besides the computation timing, we note that the memory transfer time from CPU and GPU is 0.7 ms for 256^2 data, 2.0 ms for 512^2 data, 7.5 ms for 1024^2 data. By using the prefetching scheme, a speedup ratio of 20% is achieved for the bilateral filter.

Next, the NLM filter is applied to the test dataset. Figure 12 demonstrates the image quality of the NLM filter with different window sizes. In the NLM filter, there is a parameter *h* that controls the noise reduction effect. We also find the approximately best parameters for different windows sizes. The large neighborhood size ($(11+17)^2$ in Figure 12 (a)) resulted better quality than in the smaller neighborhood case ($(11+11)^2$ in Figure 12 (b) and $(11+7)^2$ in Figure 12(c)).

Table 3. The performance of bilateral filter (in millisecond)

Image size	Neighborhood Size	Bilateral	Optimized Bilateral	Speedup
256^2	7^2	0.192	0.131	1.46
	11^2	0.309	0.246	1.25
	17^2	0.650	0.539	1.21
512^2	7^2	0.411	0.326	1.26
	11^2	0.927	0.705	1.31
	17^2	2.150	1.760	1.22
1024^2	7^2	1.446	1.120	1.29
	11^2	3.374	2.473	1.36
	17^2	8.080	6.545	1.23

The NLM filter's performance is shown in Table 4. The prefetching method resulted up to 4× speedup in this filter. This is because the NLM filter has one order of magnitude more neighborhood searching to do than the bilateral filter, which make them clearly a memory bounded problem. Our optimized filters outperformed the bilateral filter and the NLM filter implemented in the CUDA SDK 3.0 by similar speedups. The performance shows the more neighborhood lookups, the more effective the shared-memory acceleration will be. Based on the fact that the NLM filter is usually one order of magnitude slower than bilateral filter but has better denoising quality, our proposed method would make expensive neighborhood filters more practical while enjoying the superior image quality.



(a) $h = 17$ $r_p = 5$ $r_w = 8$ (b) $h = 17$ $r_p = 5$ $r_w = 5$ (c) $h = 18$ $r_p = 5$ $r_w = 3$

Figure 12. NLM filtering results.

By using single precision floating point data, the largest amount of required shared memory is $(32+2\times(8))^2\times4 = 9216$ byte for the bilateral and the NLM filter. They are below 48KB as the limit of shared memory per SMP in Nvidia's Fermi card. With the development of more advanced GPU hardware, we can expect that larger preloads such as 64×64 in 32bit floating point data will be supported in the future.

Table 4. The performance of Non-Local Mean filter

Image Size	**Neighborhood Size**	**NLM**	**Optimized NLM (millisecond)**	**Speedup**
256^2	$(7+7)^2$	8.708	2.097	4.15
	$(7+11)^2$	23.927	5.034	4.75
	$(7+17)^2$	51.095	12.703	4.022
512^2	$(7+7)^2$	31.026	7.339	4.23
	$(7+11)^2$	76.494	17.756	4.31
	$(7+17)^2$	182.497	42.066	4.34
1024^2	$(7+7)^2$	118.831	28.041	4.24
	$(7+11)^2$	292.970	67.727	4.33
	$(7+17)^2$	699.231	161	4.34

The results showed that advanced acceleration technique can further speedup straightforward GPU implementations of nearest neighborhood filters. The speedup can be up to 4 times for the large window size case, which can bring high-quality denoising filters real-time performance. The optimized de-noising filter lends itself as an independent component which can be plugged into the iterative reconstruction framework and boost the performance of the whole pipeline.

Chapter 3. Parameter Optimization

Flat panel cone-beam CT has become a major imaging technique due to its simplicity, uniform resolution, and low scanning time. The traditional cone-beam CT reconstruction method is the FDK algorithm [29], which can provide high resolution results but requires several hundreds of X-ray projections. With growing concerns about the potential risk of X-ray radiation exposure to the human body, low-dose CT has become a significant research topic ([36] [76] [96]). Dose reduction usually involves lowering the X-ray energy per projection (i.e. low mAs) and/or reducing the total number of projections. Both methods suffer from low signal-to-noise ratio (SNR) in the reconstructions. In the low mAs case, the reconstructed image is typically noisy, while in the low projection number case the result can suffer from streak artifacts. Iterative reconstruction schemes, matched with suitable regularization techniques have been shown to cope well in the adverse settings of low-dose CT. Both iterative reconstruction and regularization typically offer a diverse set of parameters that allow control over their quality and computational speed. As such they are significantly more complex to use than FDK. Given a specific imaging task, choosing the best setting of each of these parameters can be tedious and is often a matter of domain expertise and intuition. Further complicating the situation is that this expertise is both domain-specific and dose-dependent. The knowledge on which setting to pick is typically acquired through many experiments, and only domain experts have the ability to balance trade-offs that may exist among different parameters. Due to this complexity, unleashing the true potential of iterative methods for everyday clinical use is still an on-going process.

What is lacking is an automated procedure that can arrive at good parameter settings – and furthermore, enable the use of these settings for any new reconstruction within a similar scenario. With prolonged training such a system would then store a good deal of scanning expertise and could serve as an advisor to the human operator. An early attempt in this direction was a framework which used the computational power of GPUs to quickly compute the outcome of all possible parameter combinations, at some level of

discretization, and then present the optimal combination to the user ([97] [95]). While this work revealed interesting relationships among the parameters tested – there were two – such an exhaustive scheme is clearly not scalable in the number of parameters. In this work we derive such a scalable framework. It uses a specific form of genetic algorithm – ant colony optimization [22] – to navigate the high-dimensional parameter space efficiently.

Image quality and radiation dose are important factors, but when it comes to clinical practice, the time it takes to arrive at a suitable reconstructed image also plays a critical role The latter is an objective often disregarded in the literature, but it can be of great importance when CT scanning is part of a surgical procedure, in emergency departments, but also when the patient is anxiously waiting for a diagnosis. Thus altogether we are facing three co-dependent objectives – radiation dose, reconstruction quality, and computational speed (DQS). This three-tier multi-objective optimization problem has several well-known non-linear trade-offs governed by its native domain. For example, a lower radiation dose will require a higher computational effort to reach a certain reconstruction quality – but it may never reach the quality of a regular-dose reconstruction no matter how involved the computations are. This is true even for an optimal parameter configuration, and it essentially means that there are many parameter configurations that are non-competitive. This enables us to efficiently cull the search space in the optimization procedure. Essentially, we can reject any solution which, at the same quality and dose, has a lower computational speed. Furthermore, assuming similar growth patterns in quality, we can reject, for computational reasons, all solutions that at any stage of the iterative process have fallen significantly behind the others in terms of quality.

Our optimization is geared towards quality (Q) and speed (S) – dose (D) is an input factor. To demonstrate how our QS optimization works we have chosen an algebraic reconstruction framework with an interleaved regularization via non-local means (NLM) filtering [9], which has been used for regularization in the past with good success [34] [97] [109]. Further encouraging is the recent work by Li et al. [51] which makes a strong

case for NLM as a possibly superior alternative to TVM. We have further augmented the NLM-based filtering with unsharp masking for edge enhancement. . Our research advances the early work of Xu and Mueller [101] which used a standard genetic algorithm in conjunction with bilateral filtering [84] for regularization.

Our QS optimization component is quite general and could also be used to find good parameter settings for other iterative reconstruction algorithms, such as EM, and other regularization techniques, such as TVM ([75] [76] [78]), soft-threshold filtering [103] or Tight Frame (TF) regularization [36]. In fact, we show that our algorithm compares quite favorably with ASD-POCS [76]. It differs from this scheme in that it does not require a special – and time consuming – control unit to guide and discover regularization parameters during the convergence. Rather, our method is fully informed by prior knowledge and so can complete its procedure along a given path deterministically without any further probing of the current state.

As mentioned, DQS advisor adds the reconstruction speed component to the more standard DQ comparison. Tang et al. [82] performed such a DQ study on a single mAs setting and found that it is preferable to distribute total imaging dose into many view angles to reduce the streak artifacts caused by angular under-sampling. Yan et al. [102] investigated this subject with multi-mAs setting. They plotted both quality and dose as functions of number of projections and mAs per projection. Based on this insight, protocols can then be developed to maximize the dose reduction while minimizing the loss of image quality for various imaging tasks. Our DQS optimizer enables similar insight but in the context of the time needed to produce the reconstruction.

When it comes to DQS, there are always trade-offs. And if only for educational purposes, visualizing these tradeoffs in a global context can be immensely helpful to appreciate their extent. Since we have a tertiary relationship a bivariate linear graph is insufficient to capture these relationships. An interactive parameter space visualization framework for iterative CT was introduced by Xu and Mueller [99] which used icons to visualize additional variables in a 2D plot. We have chosen a different approach that requires less interaction. It plots the third variable as a backdrop layer of a 2D plot and it

allows users to click on this plot to visualize an actual reconstructed image. Upon clicking a marker is placed into the plot and the image is inserted into a comparative matrix of images. We believe that comparing actual images is ultimately the best way to appreciate quality since any error metric, RMS, CC, SSIM, CNR, and various others, never capture the full impact an image has on the human visual system. Finally, the knowledgebase through its visual front-end is exposed as a web service and can be accessed via standard web browsers. Future generations of this service will allow users to upload their own data, simulate low-dose effects, run the optimizer, and visualize the DQS.

Figure 13 illustrates the parameter optimization workflow during the training stage. The gold-standard was generated by the high dose scan, and we perform iterative reconstruction with regularization in low-dose acquisitions. The optimization was done per iteration in iterative reconstruction. Figure 14 lists the reconstruction and regularization parameters inside the iterative algorithm component.

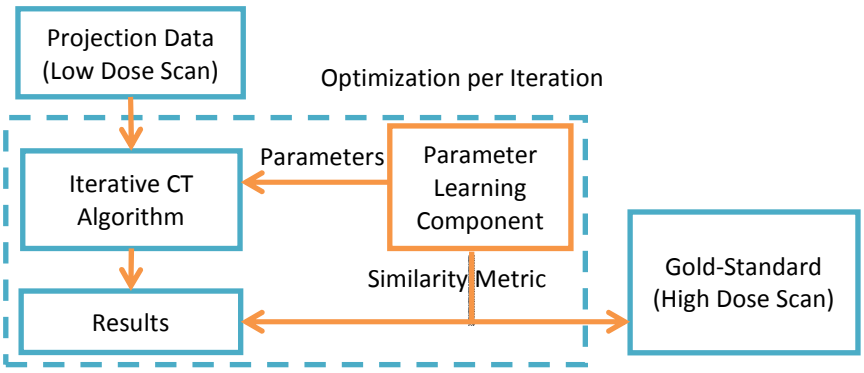


Figure 13. Parameter optimization for the training dataset.

3.1 Ant Colony Optimization (ACO)

We model iterative CT reconstruction as a path searching problem, as illustrated in Figure 15. In this graph of nodes and edges, each node represents a unique state of the volume that is being reconstructed, while each edge represents the computation of a correction pass and a regularization pass. The edge weight represents the time cost, which in our case can be uniformly set to 1 since all passes have a fixed constant cost. Each

node is tagged with a score that encodes a quality metric. The overall goal is to find the node that has the highest score within a given cost. Since all edges have the same weight, the problem is reduced to find the best score after a fixed number of steps.

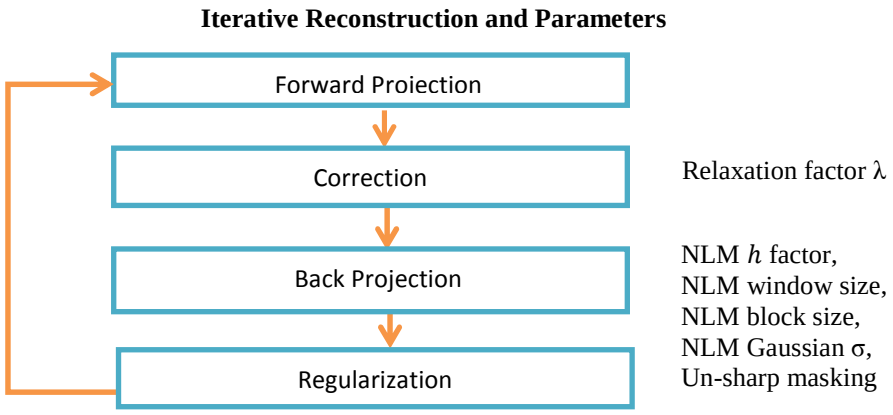


Figure 14. Iterative reconstruction and parameters.

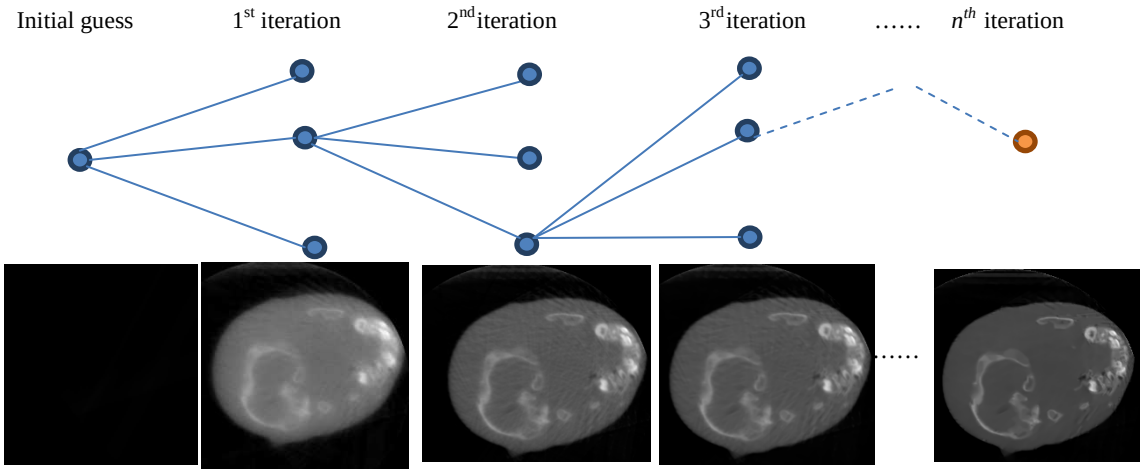


Figure 15. Ant colony optimization searches best parameters per iteration.

The ant colony optimization (ACO) algorithm [22] is a swarm intelligence method to search for good paths in discrete graphs. Intuitively, it launches a large number of artificial ants searching for the best score. Each artificial ant independently moves through the graph and receives a score based on image quality, reconstruction time, and dose. Ants with good scores have their paths reinforced with pheromones to attract other ants. The probability for an ant to choose an edge connecting two nodes is affected by the

moving trend of all ants. The probability for choosing an edge will increase if a large number of high score-ants have traversed it.

We attempt to optimize the parameters including the relaxation factor λ in Equation (4), the h factor, Gaussian blur factor, window size and block size for the NLM filter in Equation (9) and Equation (10), and the un-sharp masking factor. These parameters can be different per iteration, resulting in an astronomical search space. For example, assuming we allow 100 discretized values for each parameter and run the pipeline for 10 iterations, the search space will be 10^{120}. This search space is so huge that simple exhaustive search algorithms will fail to find the optimal solution in a reasonable amount of time.

We adopt a greedy heuristic to prune the search tree. This heuristic guides ants with a "best guess" for the path on which the solution lies. As shown in Figure 15, the greedy ant system only searches the best parameter setting for a single iteration, then adopts the best setting and moves on to the next iteration. The solution space can then be reduced from 10^{120} to 10^{13}. The pseudo code for this algorithm is shown in Figure 16.

```
1       FOR each iteration in iterative reconstruction
2            SET GENERATION_COUNT = 1
3            WHILE GENERATION_COUNT <= MAX_GENERATION
4               FOR each ant in the current generation with size ANT_GN
5                      Use prev-iteration-best-results as input
6                   Obtain a set of randomized parameters using pheromone
7                      Perform one iteration of iterative reconstruction
8                      Record the quality scores using quality metric and gold standard
9                      Update current-generation-best-results
10              ENDFOR
11              Update pheromone
12              IF ABS(current-generation-best-results - prev-generation-best-results) < ε
13                 AND current-generation-best-results > prev-iteration-best-results
14                     SET prev-iteration-best-results = current-generation-best-results
15                     GOTO LINE 19
16              ENDIF
17              SET GENERATION_COUNT = GENERATION_COUNT +1
18              SET prev-generation-best-results = current-generation-best-results
19           ENDWHILE
20      ENDFOR
```

Figure 16. The pseudo code for per iteration optimization.

For each iteration of iterative reconstruction (line 2-19), our system creates generations of artificial ants. The input for each generation (line 6-9) is the previous CT iterations result, which is initialized to zero and updated per CT iteration (line 14). Line

6-7 is the most time consuming part. In that stage, each ant first obtains their parameters probabilistically using pheromones, and then uses these parameters to perform one pass of forward projection, correction, back-projection and regularization. Then the system stores a quality score for each ant. In line 11 we update the pheromone value after a generation of ants finish their run. Line 12 and 13 are the two convergence conditions. It will make sure the optimization converges and the current iteration should have better scores than in the previous iteration, otherwise we launch another generation of ants until both conditions being met.

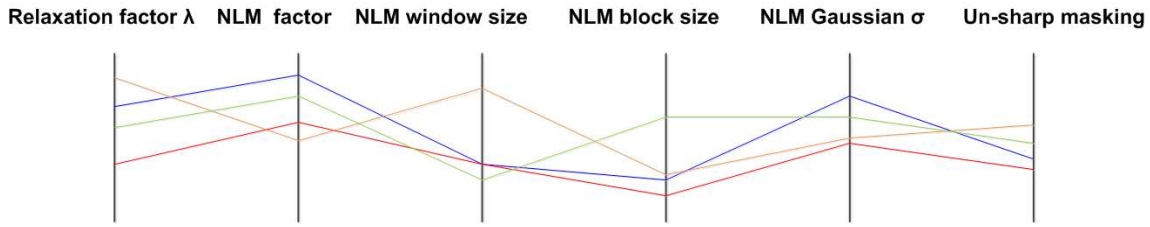


Figure 17. Different settings of parameters shown in parallel coordinate.

Figure 17 uses parallel coordinate show four sets of parameters. An ant's choice is a 6-tuple shown as a line. Ant colony algorithm launches many lines and eventually converge them into a single line through pheromone control. The pheromone for the current generation is updated for the next generation use, making ants more likely to choose parameters similar to previous best parameters. In the following, we describe the pheromone's role in detail. The probability for an ant to choose a discretized value j for ith parameter is:

$$P(i,j) = \frac{\tau_{ij}}{\sum_{q=0}^{R_i} \tau_{iq}} \tag{14}$$

where τ_{ij} is the pheromone on value j for ith parameter and R_i is the discrete resolution of the ith parameter. The value of τ_{ij} is initially set to one; this will allow the ant to make purely random decisions. After a small group of ants (in our case it is 50) finishes their moves for all 6 parameters, the pheromone is updated as:

$$\tau_{i,j}^{(m+1)} = (1-\rho)\tau_{i,j}^{(m)} + \bar{s}_{i,j} \tag{15}$$

where $s_{i,j}$ is the normalized score (with $0< s_k <1$) of an ant choose the value j for ith parameter. $\bar{s}_{i,j}$ is the average score of all the ants ant choose the value j for ith parameter. $\tau_{i,j}^{(m)}$ is the pheromone on value j for the ith parameter for the mth generation of ants. ρ is the pheromone evaporation factor (with $0< \rho <1$). The range of pheromone is clamped within [0,1]. The Equation (15) updates the pheromone such that later ants will be more likely to follow the path of previous high-score ants. In the absence of a human observer, s_k is generated by a computer based on a quality metric. In this work, we use the correlation coefficient (CC) to determine $\bar{s}_{i,j}$.

3.2 Single mA Experiments

In our experiments, we collected two sets of data from a cone-beam CT scanner (Medtronic O-arm). We perform the reconstruction algorithms on the central slice only. The detector 1D resolution was 1,024 pixels with pixel-size 0.388mm. The low-dose case was set as 72 evenly-distributed projections and the gold-standard was generated by the FDK algorithm with 360 projections. In the OS-SIRT scheme, we made 20 subsets and each subset contains 3-4 projections. Head phantoms were used to test the parameter optimization algorithm and the training results are shown in Figure 18. The result from 50 iterations of the OS-SIRT algorithm with a constant $\lambda = 1$ is in panel (a) and the gold-standard is in panel (b). 50 iterations of the ASD-POCS result is displayed in panel (c). This result is smooth but has some typical cartoon-like structures due to the TVM scheme. On the other side, our trained results (d) are the most similar to the gold standard (b) and can preserve sharper details than ASD-POCS can (c).

The CC metric was applied to central 256x256 regions to emphasis the region of interest. Figure 19 shows the plot of trained parameters through 50 iterations with CC scores. We can see that there was very little improving in CC after 40 iterations and CC stopped improving after 50 iterations. Figure 20 shows the plot of the relaxation-correction factor in Equation (4). The optimized parameter suggests starting with a bigger value around 2.0 and gradually decreasing to 0.1. Note the overall decreasing trend contains a certain amount of perturbations.

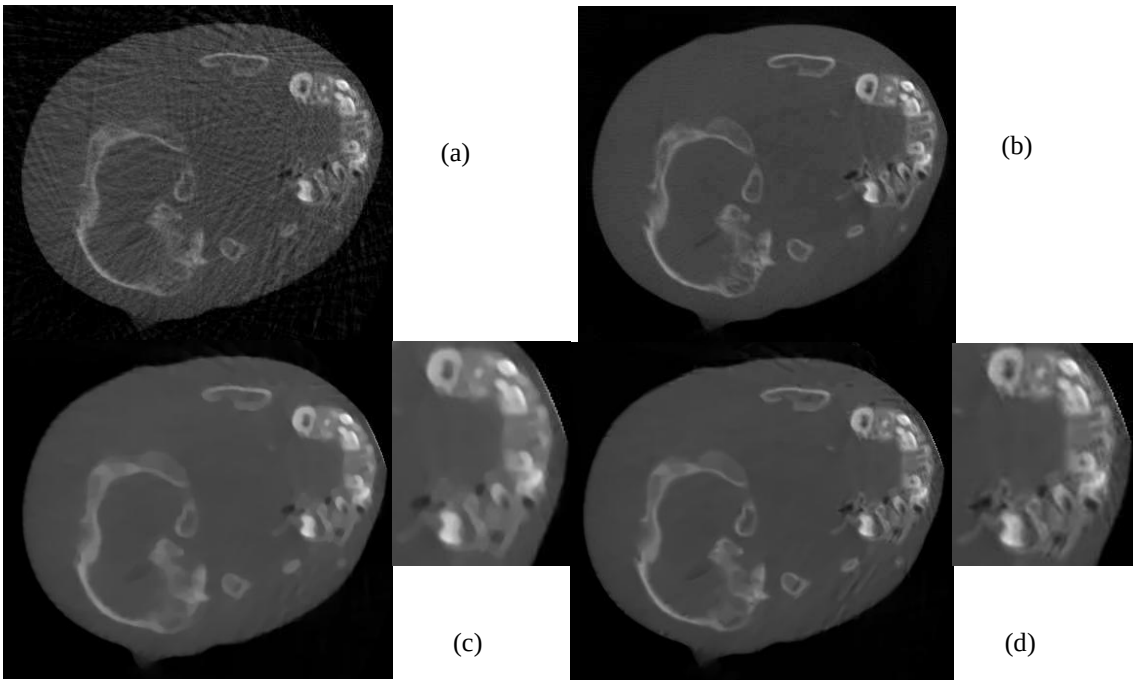


Figure 18. A central slice of a reconstruction for a training dataset. (a) low-dose OS-SIRT. (b) gold-standard FDK by 360 projections. (c) low-dose ASD-PCOS and (d) optimized low-dose. (a), (c) and (d) use 72 projections and 50 iterations.

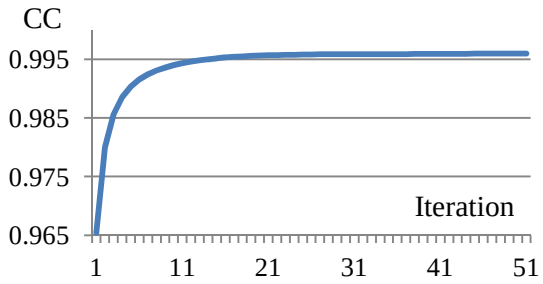


Figure 19. The Correlation-Coefficient (CC) through 60 iterations.

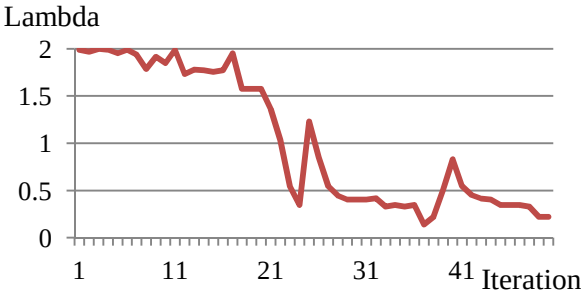


Figure 20. The relaxation factor λ through 60 iterations.

We tested the learned parameter setting on another similar head phantom. The central slice was shifted to another autonomy region in the head phantom. Figure 21 documents the image quality of the new dataset with (a) 50 iterations of OS-SIRT with a constant $\lambda = 1$, (b) 360 projection FDK, (c) 50 iterations of ASD-POCS and (d) 50 iterations of optimized parameters. We observed that even in another scan, the parameters can guide the reconstruction to achieve better visual quality than ASD-POCS.

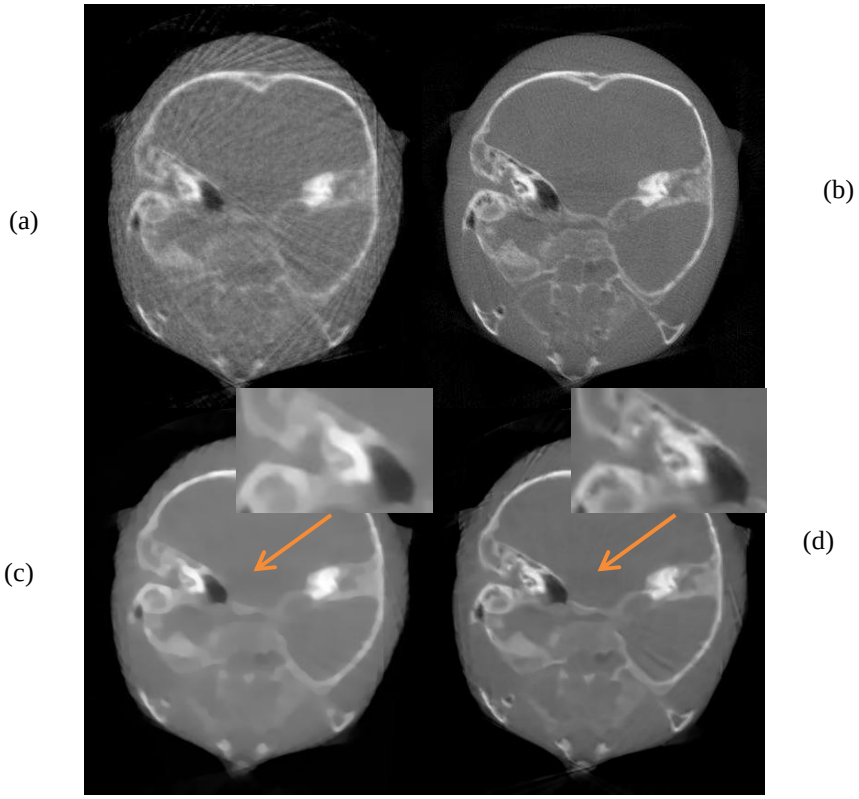


Figure 21. A central slice of a reconstruction for a new dataset. (a) low-dose OS-SIRT. (b) gold-standard FDK by 360 projections. (c) low-dose ASD-PCOS and (d) optimized low-dose. (a), (c) and (d) use 72 projections and 50 iterations.

3.3 Visualization Interface

We use our parameter optimization engine to perform optimization-guided CT reconstruction and take both quality and speed into the consideration. The optimization generates multi-dimensional data containing many fields (dose, quality, reconstruction time, parameters). We presented the interactive multi-dimensional interface as a web

service for easy access. Users can compare images from different settings and to make informed decision.

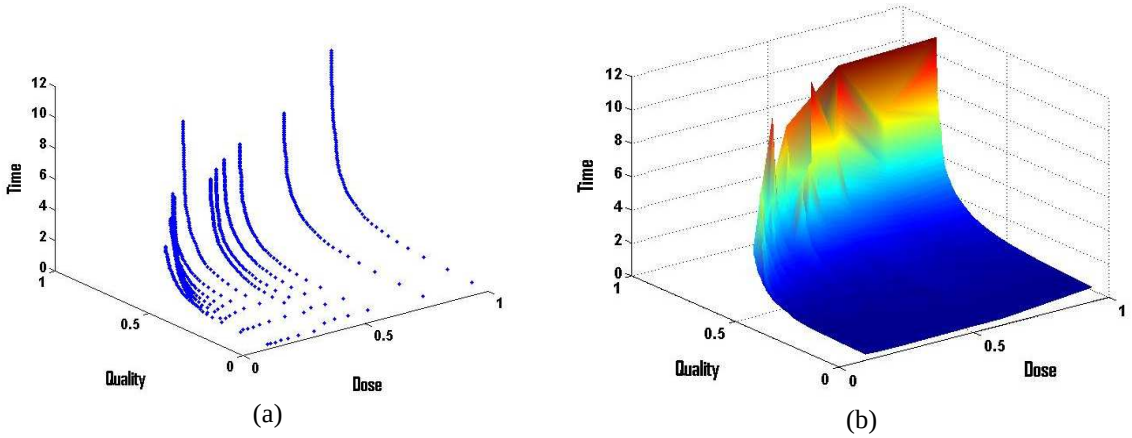
3.3.1 Visualization Design

Instead of an interactive system, a straightforward alternative method is to display several individual plots using visualization software to look at the different domains one by one. These plots can by created by many softwares, such as Excel, Matlab or Mathematica. One major disadvantage of this method is the lack of user invention during the data exploration. The images created by these general purpose software's are usually considered as static. It can not support the visual exploration, when users want try out a different visualization setting or look at the image associated with a certain data point. On the other side, the advantage of interactive system over multiple static plots is that user can have the better understanding of the big picture. Users will be able to travel in the high dimension space identify similar solutions nearby. This will help users to identify the trend of the converging.

We chose to use the continuous scatter plot technique to present the underlying data. In our problem, there are four dimensions to represent one reconstruction: X-ray current (mAs), number of projections, quality and time. The visualization system presents these four-dimension data into a single picture. We created an auxiliary axis -- dose axis as X-ray current multiplying number of projections. In Figure 22, a multi-mAs experiments data is presented by a 3D plot. Figure 22(a) is a 3D scatter plot. Dose, quality and speed are mapped into a three spatial axis respectively. Figure 22(b) shows an interpolated surface space with color showing the time axis. In comparison, the continuous scatter plot used in Figure 22(b) is more effective to convey visual information.

The data points cluster around several vertical lines for two reasons. First, our iterative reconstruction optimization engine was running per iteration. Thus the generated data shows the optimized quality and speed of the iterative reconstruction for each iteration. As the iteration number grows, the quality and time changes but the radiation dose will not, since they are still based on same acquisition data. Thus we see series points changing in Y coordinate and color but not in X coordinate. Second, X-ray tube

current and projection number only have discrete values available on the CT scanner, thus the data points will be grouped into several discrete values along dose axis.



(a) (b)

Figure 22. The input data in 3D visualization.

The remaining challenge is to embed the information about mA and number of projection. We can express Figure 22(b) by using its two orthogonal projections. Thus we have two projection images: One is a quality vs. dose plot with time mapped to color. The other one is a time vs. dose plot with quality mapped to color. Conflict happens when two data points are projected into the same 2D location. With the rest of fields being equal, the preference will go to the better speed and higher quality. This can be easily implemented by OPENGL depth-culling. We can flip the depth testing direction by setting GL_GREATER or GL_LESS. Both modes are inherently scatterplots but are triangulated in order to have smooth color interpolations for arbitrary locations. Users can have both plot available and choose the desirable mode to look at. Here we use the quality vs. dose mode as an example to describe the user interface.

3.3.2 Interface

The interface of the DQS Advisor is composed of a main plot and some control sliders and picture displayer as shown in Figure 23. The main plot is a 2D scatter plot of dose and quality with different colors to show running time. The background color of the pixel represents the nearby tuples' time field and the global colormap can be adjusted by users through a range slider at the bottom. There are two sliders along the X and Y axis

respectively. They can be jointly used to display a highlighted view of a certain region. On the top of the plot there are two lines to illustrate dose composition. By adjusting four sliders on the right, users can specify the region of the plot to be visualized. On the bottom-right there are four image slots to display reconstruction results of marked data points. Here we discuss the supported user interaction in details.

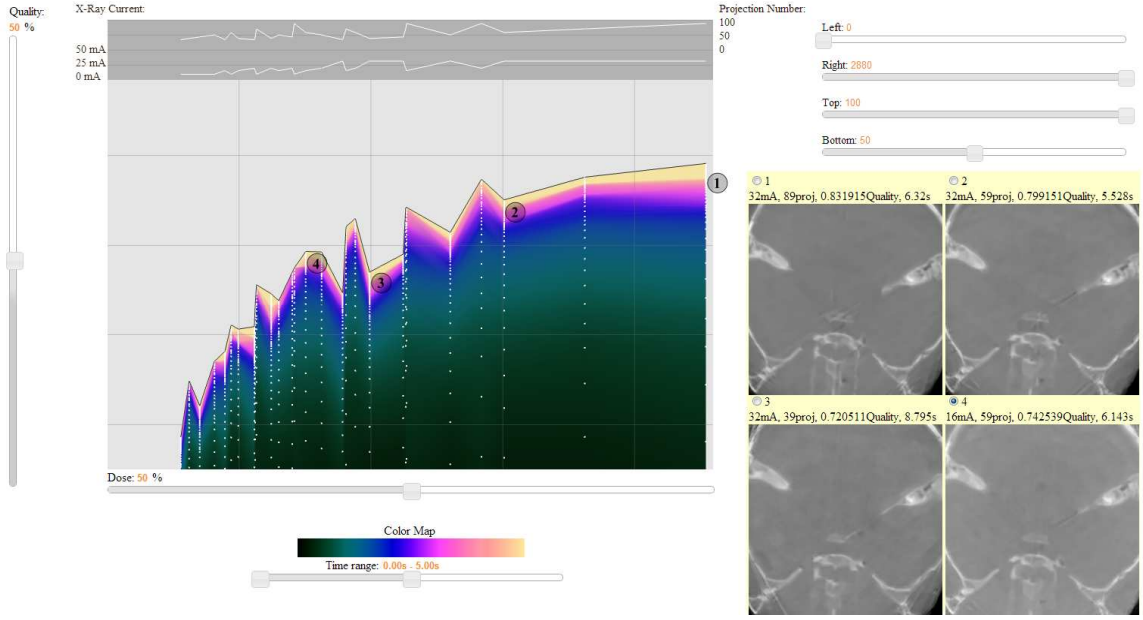


Figure 23. The quality vs. dose plot.

Main plot: The X axis shows the dose as product of mA and number of projections. The Y axis shows the measured quality in exponential scale. All data points are displayed as white points and the user has the option to enable or disable the data points.

Adjustable colormap: users can use a range slider to adjust the values to be mapped to the color spectrum. The color spectrum is designed as static, from black on one side to yellow on the other side. This colormap is placed below of the plot and is shown in Figure 24. Users can dynamically adjust the upper/lower values to be mapped to the black and yellow respectively. As shown in Figure 24, panel (a) shows the default color-coded plot and panel (b) shows the updated plot after adjusting upper bar of the range slider. This help user to identify the pattern of the mapped value for different regions .

Dose representation: dose is represented as mA multiplying number of projections. In this way, two dimensions are compressed into one scalar value, represented in the X

axis. To present the missing value of the mA and number of projections, we display two lines on the top region of plot. These two lines represented the mA and number of projections respectively, showing the different composition to a same total dose, as shown Figure 25. For example, the dose level indicated by the blue vertical line is made from 16 mA and 89 projections.

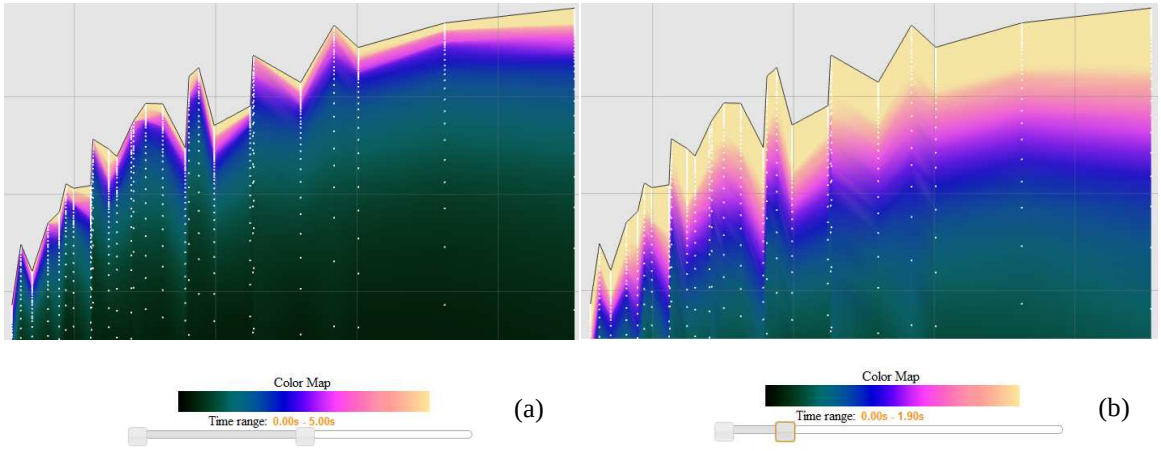


Figure 24. Changing the colormap.

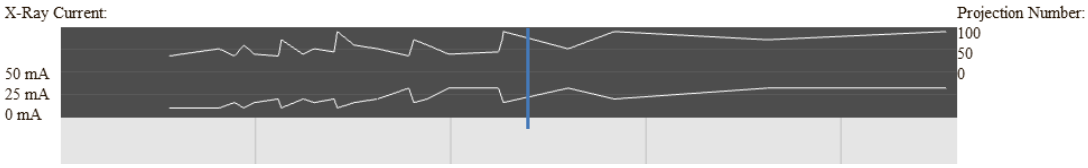


Figure 25. Displaying the mA and projection number.

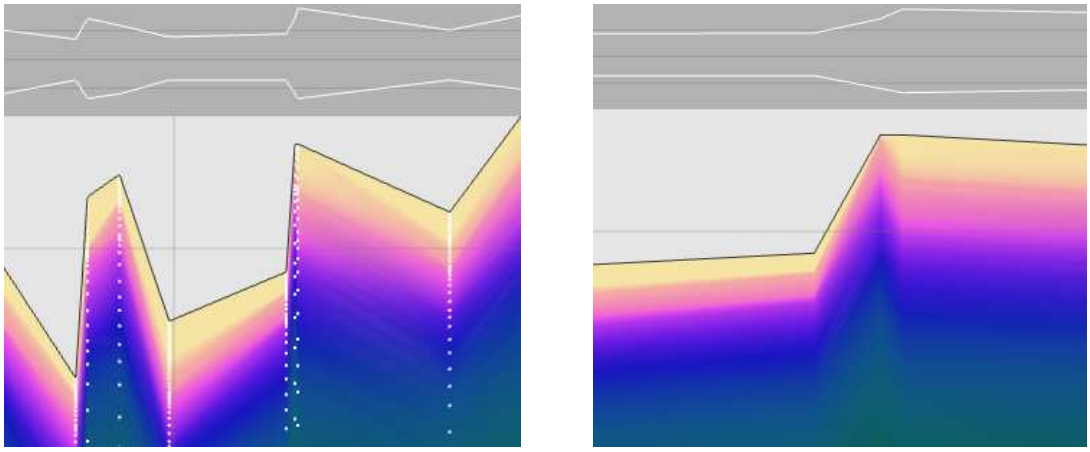


Figure 26. Changing the region of interest.

Region of interest: users could choose to better plot the results by modifying the left/right/top/bottom boundary of the main plot. In this way, users can pan or zoom in/out to inspect the plot. This functionality can help users to explore the sparse space. The zoom in/out is a linear function that will preserve the neighbourhood without extortion. Figure 26 shows two detailed visualizations after adjusting the region of interest. In panel (a), a small region containing 80+ points are displayed in a zoomed in manner. These details can not be captured well without the help of region of interest visualization. Panel (b) shows another region containing color interpolation with the data points disabled. We see that the background color can be better visualized without data points.

Image Displayer: Users can mark up to four data-points and the corresponding images will be displayed for side-by-side comparison. The data information will also be displayed. This function is based on left click. Whenever a left click occur, the system will find the nearest data point around the cursor. The nearest neighbor search is accelerated on an R-tree structure which will be discussed in Chapter 3.3.1. In the meantime, a numbered icon will be marked on the clicked location. In Figure 27, four images were put side by side for comparison with settings on the top. The first image was reconstructed from 32mA and 89 projections with quality score 0.8319 and speed 6.32 seconds. The last image was reconstructed from 16mA and 59 projections with quality score 0.7425 and speed 6.14 seconds. In this way, users can understand the image quality much better than plain numbers.

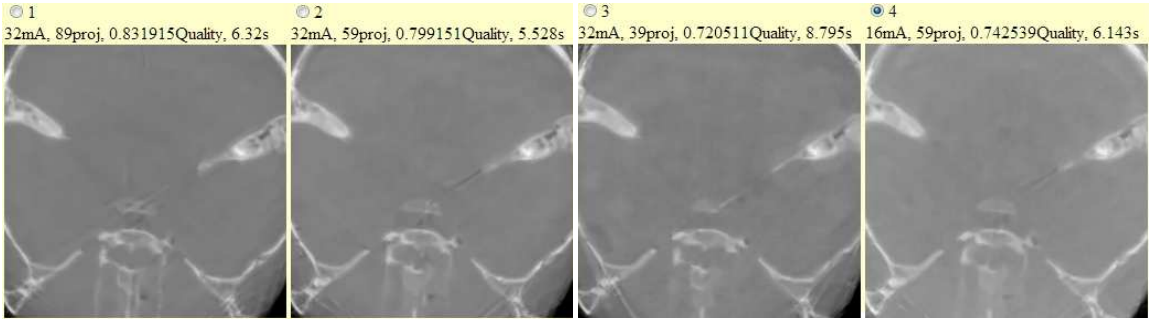


Figure 27. A side-by-side images comparison.

Highlighting slider: When enabled, user can spotlight a certain 2D neighbour using a vertical slider and a horizontal slider. Other un-highlighted region will be gradually color-blended.

Right click menu: A popup menu will show up when a right click occur. User can set preference on some display mode. User can also display/hide the data points, horizontal lines, vertical lines, highlighting slider and grayscale color.

3.3.3 Scalability

We host our visualization tool on a Web Server and can be accessed publicly at http://vail.cewit.stonybrook.edu/Projects/CTPlot/. The webpage is based on WebGL and JavaScript. Users are required to have a WebGL supported browser (such as Chrome and Firefox) installed. In this way, out platform enjoy the worldwide access and cross-platform support. We carefully designed the system to be capable of interactively displaying relatively large data sets. A typical desktop browser could interactively manage 2.2k data points and 2.2k 512x512 images.

Data input: The input data is consisted of 4-tuples. The four fields of the tuples are ordered as mA, projection number, quality and time. The tuple are stored in a CSV file. The images associated with data points are stored in a folder. The name format of the images are the same as the 4-tuple. We use JPEG format to store the compressed images to minimize the storage and data transfer latency.

R-tree: A naïve search algorithm will take linear search time which does not perform well with large dataset. The running time is $O(n)$ if there are n data points. We used a non-recursive R-tree to facilitate interactive 2D search. The R-tree's nearest neighbour searching time is $O(logn)$.

3.4 Multi-mA Experiments

In this experiment, we performed training across different mA setting and projection number. The aim of this experiment was to create a desirable scan protocol with the help of the DQS advisor. We collected four sets of data from Medtronic O-arm system. They are under 10mA, 16mA, 20mA and 32 mA respectively. The exposure time for each

projection is 10ms. So for a single projection the mAs value will be the mA setting multiplying 0.01. Under each mA setting, we selected evenly distributed projections according the separation angle. The X-ray angle separations used were 4, 5, 6, 7, 8, 9 and 10 degrees. The subset number in OS-SIRT was 20. A head phantom was used in this experiment and we optimized the central slice reconstruction.

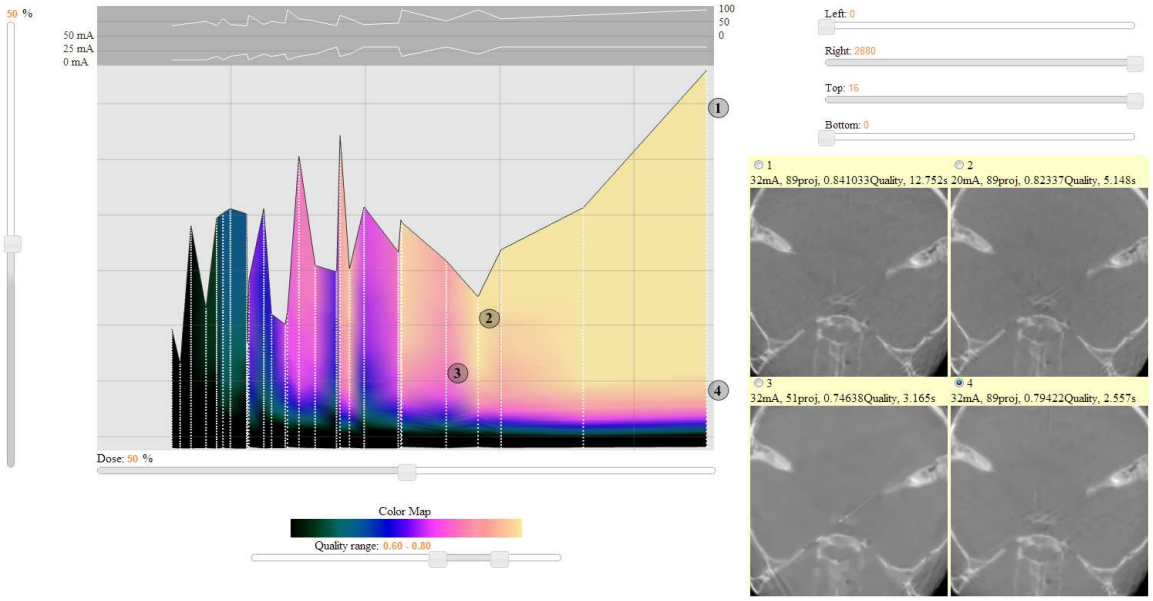


Figure 28. The speed vs. dose plot.

The training results are display as a web service as previously shown in Figure 23 described in Chapter 3.3.2. Here we look at the other mode: the speed vs. dose plot as shown in Figure 28. Similarly, the X axis showed the product of mA and number of projections. The Y axis shows the time spent on reconstruction. The quality is color-coded and we apply triangulation to interpolate the color in 2D space. The interface assists users to quickly find the preferred acquisition settings under the input constraints. It provides a dynamic view of subtle parameter changes by moving the control bars so that users could be aware of the impact of some specific parameter.

The results confirm the result of the recent study conducted by Yan et al. [102]. By observing the dose composition and image quality in Figure 23 and Figure 28, we can easily identify the positive correlation between number of projection and maximum quality. The projection number affects the image quality more than X-ray current. With

similar overall dose, higher projection number case usually can reach better quality than higher mA case. The possible explanation is that fewer projection create streak artifacts. Compared to the Gaussian noise created by a lower X-ray current, streak artifacts are more likely to be treated as a feature in the regularization algorithm and thus more difficult to remove.

Chapter 4. Verifiable CT Visualization

In this chapter we concentrate on error propagation in volume rendering of the data acquitted by CT scanner. We propose a Verifiable Direct Volume Rendering method which is able to certify a CT reconstructed volume for use with a given interpolation filter, explicitly specifying the maximum error that might occur in the rendering. It uses a mixed-resolution volume encoding computed in a pre-process. This representation can maintain the advantages of real-time rendering, and at the same time gives the verifiable results.

4.1 Volume Rendering

Volume rendering includes a wide range of techniques. Engel et al. give an overview of real-time volume rendering methods [23]. The volume rendering technique can be classified as indirect method, such as iso-surface reconstruction and direct method that immediately display the voxel data. Compared with indirect rendering techniques (e.g. the Marching Cube algorithm [58]) using surface with geometric primitives, Direct Volume Rendering (DVR) is a more attractive approach. It is more accurate since the rendering samples are given by direct interpolation without any intermediate representation.

DVR as a large category, contain several methods. For example, ray casting, cell projection, shear-warp, splatting and texture-based methods. The common theme is an approximate evaluation of the volume rendering integral for each pixel. The light intensity I is calculated through integration along the ray. The continuous integration along the ray $x(t)$ can be computed through

$$I = \int_0^D c(x(t)) \exp(-\int_0^t \tau(x(t'))dt')dt \tag{16}$$

where D is the length of the ray, $x(t)$ represent the scalar value of volumetric field sampled the along the ray, $c(v)$ is the color transfer function and $\tau(v)$ is the opacity

transfer function representing attenuation coefficient. The volume rendering integral is approximated by Riemann sum:

$$I \approx \sum_{i=0}^{D/\Delta t} (c(x(i\Delta t))\Delta t \times \prod_{j=0}^{i-1} \exp(-\tau(x(j\Delta t)\Delta t))) \tag{17}$$

By introducing the opacity value $\alpha_i = 1 - \exp(-\tau(x(i\Delta t))\Delta t)$ and define $C_i = c(x(i\Delta t))\Delta t$, then the equation can be simplified as:

$$I \approx \sum_{i=0}^{n} C_i \prod_{j=0}^{i-1} (1-\alpha_i) \tag{18}$$

The computation can be formulated as front-to-back composition during the ray tracing. To perform this front-to-back composition requires the use of 3D texture supported in graphics hardware to produce viewport-aligned slices. This will reduce one stage of resampling and reduce some artifacts.

In volume rendering, material classification is often done via transfer functions. The traditional 1D transfer function is based on scalar values only. Recent research has investigated a plurality of new transfer function domains which have been used together with scalar values and results from these are very promising. They include gradient magnitude [45], curvature [43], features size [13], occlusion spectrum [14] and visibility [14]. Perception can be also added into the transfer function design [11]. Mai et al. [56] presented a semi-automatic 2D transfer function design method based on segmented data. These user-controlled or semi-automatic transfer functions assume a given viewpoint. There are works on designing transfer function based on feature clustering. Sereda et al. [73] proposed to use clustering to design transfer function. Maciejewski et al. [55] proposed feature detection in 2D transfer function space automatically or semi-automatically. We focus on using clustering in the context of viewpoint suggestion.

To enrich the volume rendering images, focus+context techniques are widely used to enhance the volume rendering. Wang et al. [89] introduced the magnification lens into volume rendering. Viola et al. [88] proposed an automatic cut-away view based on assigned importance weight on segmentation. Krüger et al. devised the ClearView [47] system using spherical hot-spots based on discrete curvature based importance. There is

also research on adding multiple view information in a single view. Kohlmann et al. [46] presented a deformed viewing sphere based on history. Sudarsanam et al. [75] proposed a widget to incorporate multiple views into a single image.

4.2 Verifiable Direct Volume Rendering (VDVR)

CT reconstruction provides a major source of volume rendering. The raw scalar field is naturally available from the CT acquisition process and therefore a verifiable visualization pipeline must integrate the rendering stages with the scalar field generation. The recent insightful volume rendering approach by Rautek et al. [69] recognized this important relationship. Their algorithm, termed Direct DVR (D^2VR), integrates the raw data transformation (the CT reconstruction) stage and the rendering stage by directly generating the samples required for rendering from the raw data (X-ray projections) in place. In other words, they generate a transient volume dataset that does not require any further interpolation in volume space. They convincingly show that this has great potential for improving rendering quality. Similarly, the CT community also noticed these resampling issues that motivated D^2VR, suggesting the same direct pipeline [47].

As past efforts show, directly visualizing CT raw data is an expensive operation, since it neglects the inherent advantages of CT reconstruction, that is, the spatial coherence and data compression it provides. First, for standard DVR to render an image of n^2 pixels one requires $O(n^3)$ off-grid sample interpolations in volume space (to generate the densities along the rendering rays). On the other hand, assuming $O(n)$ projections, for D^2VR to generate $O(n^3)$ volume samples one requires $O(n^4)$ projection data interpolations. Translating these complexity arguments into practice, this causes D^2VR to be about 50 times slower than DVR, assuming tri-linear and bi-linear interpolation for DVR and D^2VR respectively. In fact, this led the CT D^2VR researchers to only develop a real-time 2D slice-viewer, shying away from volume rendering arguing the lack of sufficient computing power. Second, the CT-reconstructed volume grid also avoids the poor locality of the projection data when mapping spatially coherent data access requests. The texture fetching pattern of D^2VR in volume rendering is a *sine* function (also called the *sinogram*) which does not map well into a GPU rendering

pipeline. We can observe this from the results obtained by Xu and Mueller [93]. Their GPU-accelerated D^2VR only achieved speedups of 1.3-1.5 after applying a number of acceleration schemes, such as occlusion culling and empty-space skipping.

In order to bring verifiable visualization into practice, we propose an integrated rendering solution we call Verifiable DVR, or VDVR. Our verification procedure bridges the currently existing disconnect between the raw projection data and their visualization via volume rendering. It helps to make visualizations verifiable since we guarantee a pre-set error tolerance that is applied in the CT reconstruction step.

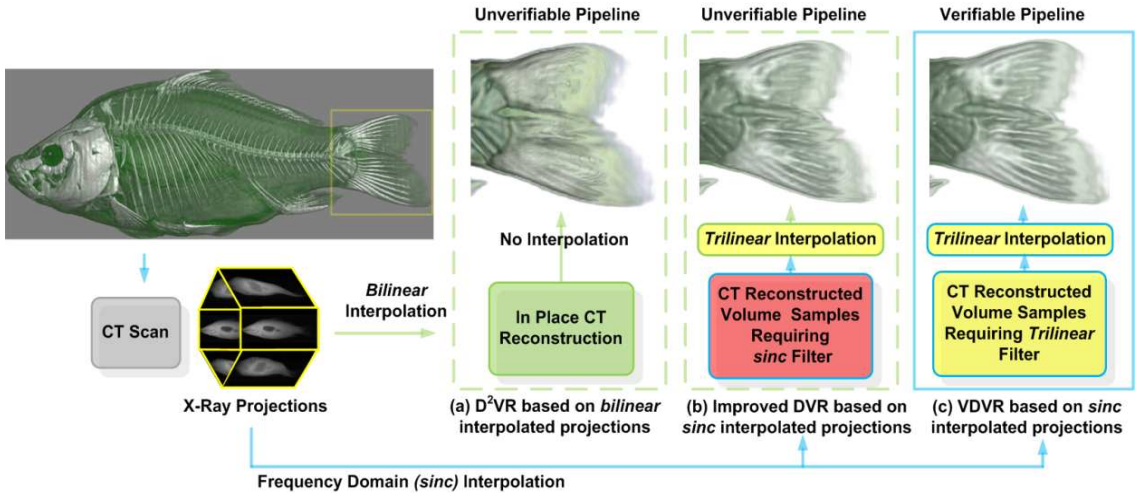


Figure 29. CT data acquisition, reconstruction and visualization pipelines for (a) D2VR [69], (b) unverifiable, (c) our verifiable method.

The fundamental difference between D^2VR, standard DVR and our VDVR is illustrated in Figure 29. Apart from the nature of the interpolated data (raw projection data for D^2VR, volume data for DVR and VDVR), the differences stem from the filter used for the interpolation. D^2VR, shown in panel (a), interpolates the projection data using a bilinear filter rather than the ideal *sinc* filter. Thus, although D^2VR effectively eliminates sampling errors in the volume domain, it still commits errors in the projections domain, which are not explicitly verified (we will show later that such a verification would lead to a prohibitively inefficient D^2VR algorithm). We can observe the resulting fidelity losses especially for the fine details on the fish tail. Conversely, both standard DVR and VDVR can effectively use *sinc*-interpolated projections, since the CT reconstruction is only a one-time process. However, volumes used in standard DVR

typically are not generated with the sample interpolation errors in mind, and so they must use an ideal *sinc* filter to make guarantees on accuracy. Therefore, when a more practical trilinear interpolation filter is used instead, fine details cannot be preserved which is evidenced in panel (b). On the other hand, a renderer presented with a VDVR-certified volume may safely use the interpolation filter for which the volume has been verified (we demonstrate this with the trilinear filter to show the gains in speed that can be obtained). This enables more details to be recovered in the rendering, as is evidenced in panel (c).

4.3 Frequency Domain Projection Upsampling

In FBP, after ramp-filtering, the projections are stored as discrete samples, to be interpolated in the later back-projection stage. Here, bi-linear interpolation is mostly used in GPU-based CT reconstruction for its fast speed performance. But this inexpensive filter cause artifacts which we would like to avoid.

The general mindset of our approach is to provide a sufficient amount of up/oversampling to allow for a verifiable approximation of the underlying continuous function by a piecewise linear function, which we can then interpolate by means of a linear filter. This mindset applies to both the projection domain and later to the volume domain. Our first task is to provide such a faithful upsampling for the projection data. As mentioned, frequency domain upsampling is the most appropriate solution for this. It is equivalent to using an ideal *sinc* filter, but without the high cost of its infinite support. In a 2D Fourier transform, a signal given in frequency space can be up-sampled by padding zeros at both ends of the spectrum (used, for example, in [57] to improve the quality of Fourier volume rendering). One can then convert the obtained signal back into signal space at the new (higher) resolution. A potential issue in frequency domain-based upsampling are sharp boundaries that may exist in the signal. Discontinuities at these boundaries introduce high frequencies. These high frequencies can be avoided by a mirror extension, which ensures a smooth transition at the boundaries [3]. Alternatively, one can also use spatial-domain windowing [50].

More concretely, our goal is a high-quality upsampling of the ramp-filtered data. Let us denote the signal as S, the upsampling operator as U and the ramp filter as F. Since the

operators are linear we may either perform $S \otimes U \otimes F$ or $S \otimes F \otimes U$. If we use filters other than the *sinc* either option will contain strong aliasing, since the affected high frequency bands are magnified by the high-pass filtering of F. It is straightforward to either integrate on-the-fly *sinc*-interpolation or prior frequency domain upsampling into the VDVR pipeline. It merely forms a pre-processing stage and the additional computation or samples will only be needed in the CT reconstruction. On the other hand, D^2VR cannot incorporate on-the-fly *sinc*-filtering or pre-computed upsampling easily. For the former the computational overhead would be prohibitive, and for the latter a 64× storage increase in GPU memory (for 8× upsampling) would be challenging. Therefore they do not perform any projection upsampling.

The upsampling process is plugged into the CT reconstruction pipeline as follows. The inputs are the 2D X-ray projections obtained from the CT scanner (or obtained with a high-quality raycaster used in our simulations). For each projection, a 2D FFT is obtained, zero-padded, and a 2D inverse FFT is run. Following, we perform a 1D FFT for each line, ramp-filter, and do a 1D inverse FFT.

4.4 Interpolation Error Assessment

4.4.1 Error Assessment for the Linear Filter

For the linear filter the largest error occurs at the local peak or valley where the maximum curvature is located [77]. Figure 30(a) illustrates this scenario for a single frequency, where the largest error occurs around the sine function's peak. Given the sampling distance d, the max absolute error E_i for a specific wavelength T_i with amplitude A_i is:

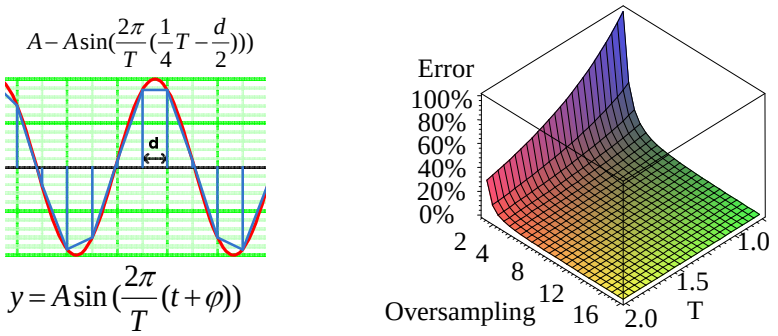
$$E_i = |A_i|(1-\sin(\frac{2\pi}{T_i}(\frac{1}{4}T_i - \frac{d}{2}))) \quad d < T_i/2 \tag{19}$$

where the maximum interpolation error E_i is a function of the sampling distance d and the signal period T_i. The distance d and period T_i are connected by the oversampling rate. If $d = T_i/2$, the sampling rate is just below the Nyquist sampling rate. In this case, the maximum error for linear interpolation could be 100% of the sine peak value. If $d =$

$T_i/16$ (equivalent to an 8× oversampling rate), this will guarantee that the error is less than 1.92% of the maximum (peak) value. Figure 30(b) illustrates the error as a function of oversampling rate and signal frequency. The error decreases with increasing oversampling rate and/or decreasing signal frequency.

Figure 30. *Sine* signal reconstructed with linear interpolation and its error.

Of course, a signal is a composite of multiple frequencies and therefore these errors



(a) (b)

would possibly compound. However, most likely these frequencies would be phase shifted which would reduce the local curvature and thus alleviate the error. Given a set of A_i, the largest error occurs when all sine peaks accumulate in one point. The largest error for this composite signal is:

$$E_{\max} \le \sum_{i=0}^{N-1} |A_i| \max\{\frac{E_i}{|A_i|}\} = \sum_{i=0}^{N-1} |A_i| (1 - \sin(\frac{2\pi}{T_i}(\frac{1}{4}T_i - \frac{d}{2}))) \qquad (20)$$

where N is the length of the signal which is also the number of frequencies obtained with the FFT.

We give a general impression on what this means in practice. In Figure 30(a) the blue curve shows the frequency amplitude for the central line of the X-ray projection of the carp dataset, while the red curve shows these amplitudes after ramp-filtering. For the blue curve, we see that the highest amplitudes are located at the low frequencies (left). The panel (b) shows the errors as a function of frequency and oversampling. We observe that for 1× resolution the highest error (100%) is located at the highest frequencies and falls off according to Equation (19), while for 8× resolution the errors all stay below 2%. Next, panel (c) shows the product of the amplitudes (the blue curve) and the error map.

The plot shows the error at the traditional resolution on a scale from 0 to 0.006 (1.0 is the maximum). Summing the errors for the full amplitude spectrum will amount to 22% of the maximum scalar value of the carp. Conversely, the errors for the 8× oversampling case are reduced by almost two orders of magnitude, and their sum is 0.4% of the maximum scalar value. We thus conclude that the residual's overall impact is negligible. Furthermore, for the filtered signal (red curve), the sum is even less.

As mentioned, the amplitude-based theoretical error assuming the worst phase shift is too conservative and likely impractical to use. Another error can be derived by taking phase shift into consideration. Let M_2 be the maximum absolute value of the curvature. Then the maximum error for the sampling distance d is

$$E_{\max} \le \frac{M_2}{2}\left(\frac{d}{2}\right)^2 = \frac{M_2}{8} d^2 \tag{21}$$

The proof [77] of Equation (21) is based on Taylor's Theorem. In general, the amplitude-based error bound is tighter for single frequency signals and the curvature-based error bound is tighter for compound frequency signals. We can use both to estimate the error.

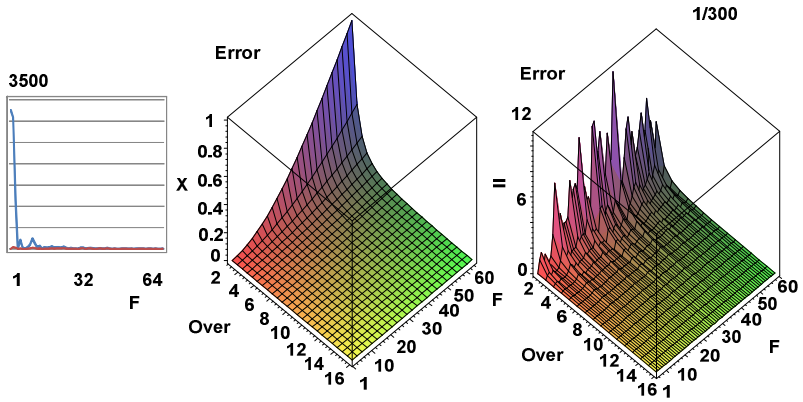
4.4.2 Error Assessment for the Bilinear Filter

For the 2D case, the amplitude-based error bound is

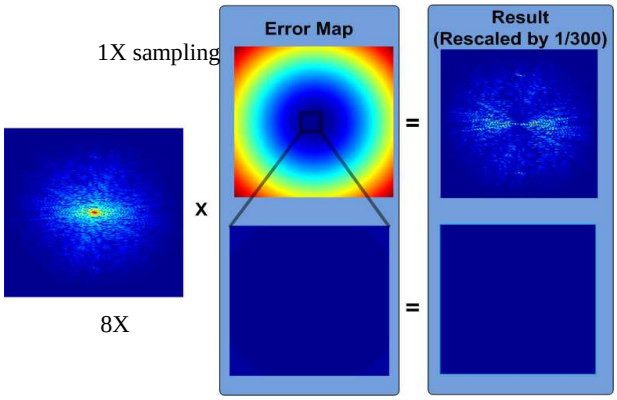
$$E_{\max} \le \sum_{j=0}^{M-1}\sum_{i=0}^{N-1} |A_{i,j}| \max\left\{\frac{E_{i,j}}{|A_{i,j}|}\right\} \tag{22}$$

where N and M are the width and height of the signal as a 2D image. Since there is no straightforward analytical solution, we computed the resulting 2D percentage-error map by extensive exhaustive search. For a given frequency $f_{i,j}$,we generated 2D sine waves with a uniform distribution of phase shifts φ and then find the maximum error inside a bilinearly interpolated unit square.

Figure 31(d-f) show the results for this 2D analysis, for a 2D X-ray projection of the carp dataset. We observe similar effects than in the 1D case. Panel (d) shows the image's amplitude spectrum using a log-scaled colormap. Panel (e) shows the error for the regular 1× sampling case (top) and the 8× oversampling case (bottom). Finally, panel (f) shows the errors multiplied by the amplitude spectrum. We observe that 8× oversampling removes the error almost completely.



(a) spectrum of a 1D line (b) percent error per fi (c) abs error per f_i



(d) spectrum of a 2D image (e) percent error per fi,j (f) abs error per fi,j

Figure 31. Linear interpolation error for oversampling. (a-c) show the largest errors for frequencies in 1D and (d-f) show the largest error for frequencies in 2D. (a) shows one half of the spectrum obtained by FFT since the other half is symmetric.

An important observation we can make in Figure 31(e) is that the 8× sampling error map is a direct copy of the 1/8×1/8 center square in the 1× sampling error map (see the illustration linking the top and bottom plots of panel (e)). So the higher degree of

oversampling, the more one zooms into the 1× error map. Also, since the raw data (the sinogram) is band-limited, the amplitude spectrum of the reconstructed signal stays constant but the overall spectrum bandwidth grows at the rate of the oversampling. Both of these facts have important implications for the maximally needed resolution of the verifiable grid, as we shall see soon.

The curvature-based error bound can be derived similarly for the bilinear filter. We use this notation later although it is not exactly the definition of curvature. In [106], we proves an error bound based on the local second order Taylor expansion which is similar to [77] [20]. Assuming the signal f is of class C^3, let M_{x^2}, M_{y^2}, M_{x^2y} and M_{xy^2} be the largest absolute values of f_{x^2}, f_{y^2}, f_{x^2y} and f_{xy^2} respectively. An error bound for the sampling distance d is

$$E_{\max} \le \frac{d^2}{8}(M_{x^2} + M_{y^2}) + \frac{d^3}{4}(M_{x^2y} + M_{xy^2}) \tag{23}$$

We can use this error bound in similar ways as in the 1D case.

4.4.3 Error Assessment for the Trilinear Filter

If N, M and L are the size of the 3D signal, the maximum error is

$$E_{\max} \le \sum_{k=0}^{L-1}\sum_{j=0}^{M-1}\sum_{i=0}^{N-1} |A_{i,j,k}| \max\{\frac{E_{i,j,k}}{|A_{i,j,k}|}\} \tag{24}$$

Similarly to the 2D case, we compute the 3D error map in 3D by exhaustive search. For a sine function with frequency $f_{i,j,k}$, we test a uniform distribution of phase shifts and measure for each the error inside a trilinear interpolated unit cube. Also similar to the 2D case, the oversampling error map can be obtained by extracting the center cube of the standard 1× error map volume and expanding it. Then given all $A_{i,j,k}$ (the 3D amplitude spectrum), we multiply this spectrum by the error volume and compute the sum, which will give us the error bound at the worse phase shift constellation.

For the trilinear filter, the curvature-based error bound can also be derived [106]. If M_α is the max-absolute value of f_α and $\in \{x^2, y^2, z^2, x^2y, xy^2, y^2z, yz^2, x^2z, xz^2, xyz\}$, an error bound for the sampling distance d is

$$E_{\max} \le \frac{d^2}{8}(M_{x^2} + M_{y^2} + M_{z^2}) + \frac{d^3}{4}(M_{x^2y} + M_{xy^2} + M_{y^2z} + M_{yz^2} + M_{x^2z} + M_{xz^2} + 3M_{xyz}) \quad (25)$$

4.5 Error Control for the Verification

Our aim is to use linear interpolation within our verifiable visualization method and to use FBP for CT reconstruction. This requires us to formally quantify the errors incurred with FBP using bilinear or trilinear interpolations (note that other constellations are possible but would have to be formally evaluated as well). There are two sources of error: (i) the interpolation of the 2D projection (raw) data during CT reconstruction of the verifiable volume and (ii) the interpolation of this volume during rendering. We discuss these two errors next.

Error control in 2D projection interpolation: In FBP, the ramp-filtering is performed in the frequency domain so no error is incurred at this stage. Then, following Equation (2), the filtered projections are interpolated and the values summed, multiplied by π and divided by the number of projections K. Directly applying the bilinear interpolation error on the filtered projections will give us K errors (one error per projection). However, we are interested in how these errors are reflected in the 3D scalar field. Essentially, this error bound is the sum of all maximum errors for the K projections, multiplied (normalized) by π/K.

Error control in 3D volume interpolation: After FBP, the reconstructed 3D scalar field is sampled and stored as a volume array of discrete samples. Thanks to the important fact that the 3D signal is band-limited we can perform the analysis on a volume reconstructed at Nyquist resolution (1× oversampling). Based on this reconstruction we can then estimate the error bound for any oversampling rate, for both amplitude and curvature-based error. In the carp dataset, assuming 8× sampling, the amplitude-based bound is 0.4 and the curvature-based error bound is 0.01 (1 is the maximum scalar value).

Flat-panel detectors produce cone-beam data. Our method extends quite naturally to this case. First, the corresponding reconstruction algorithm would be the FDK algorithm [29] [94]. While the max errors for each 2D projection can be analyzed similarly, depth-

weighting factors are now involved in the backprojection. These depth-weighting factors will multiply the projected values and so amplify the errors. Therefore the resulting error bound will be the sum of the max errors multiplied by the max depth-weight factor. The rest of the error analysis factors in the beam geometry but is principally the same as for the parallel beam case.

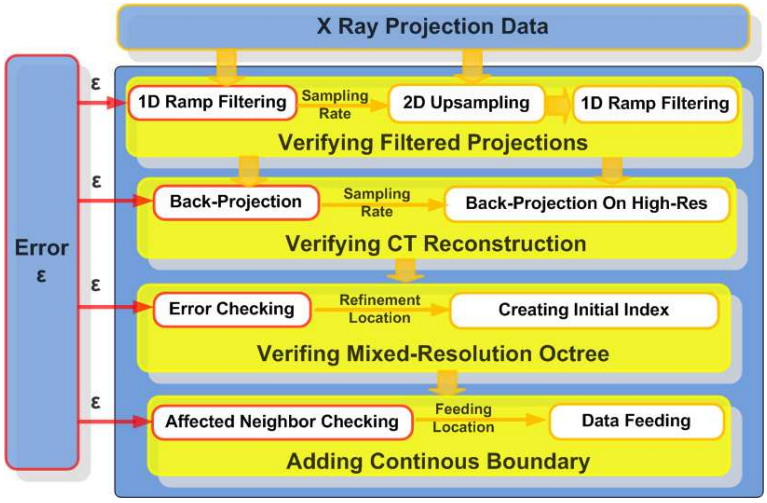


Figure 32. VDVR pipeline.

The entire verifiable pipeline is shown in Figure 32. The inputs are the X-ray projections and an error threshold ε. We first determine the upsampling rate of the filtered projections by estimating the error bound. This analysis is based on the projections after ramp-filtering. Then we run frequency domain upsampling according to the verified upsampling rate and ramp-filter the upsampled projections. Following, we perform a CT reconstruction at the Nyquist resolution (1× up-sampling), perform the error analysis and determine the ε-verified oversampling rate for the 3D volume. Then we perform back-projection again but now on a high-resolution grid which captures all possible details. We call this the gold-standard. To keep within the memory limit, we generate the gold standard in blocks of multiple cells. Within each such block, and from the gold-standard, we then build the mixed-resolution representation only keeping the detail needed. Starting from the typical base resolution commonly used, we classify those cells as subdivision cells which contain finer details. These cells are then represented

with more data points. Finally, any potential T-junctions in the mixed-resolution data are removed. In the following, we describe the first two error control processes.

4.5.1 Verifying the Projections

For the amplitude-based error bound, the maximum error for a filtered projection is the sum of its frequency errors. Then it is error multiplied by $\pi/$ to yield the final reconstruction error. We compute the curvature-based error bound in the frequency domain as well. Taking the derivatives of a signal in the spatial domain corresponds to multiplying it by a unit ramp function (the radial frequency $j\omega$) in the frequency domain [8]. Thus, if the amplitude spectrum is multiplied by $j\omega$ the IFFT will reconstruct the analytical derivative at the grid points. We use this approach to a compute a set of images, one for each derivative specified in Equation (23). We then find the maximum values in each and compute the error according to Equation (23). The reconstruction error is then the sum of the K errors for the K filtered projections, multiplied by π/K.

If both error bounds are larger than ε, this means we need to increase the upsampling rate. For the amplitude based error, we replace the error map with the one for a 2-times higher sampling rate. The curvature-based error can be simply re-evaluated. We continue this iterative process until the error is below ε.

4.5.2 Verifying the Gold-Standard Volume

The error of the gold-standard volume can be estimated also by ways of these two methods. Both use back-projection to reconstruct a volume at the traditional resolution. For the amplitude-based error bound, we take a 3D FFT, multiply the spectrum by the 3D error map for a certain oversampling rate and sum the errors. For the curvature-based error bound, we use Equation (25) and estimate the maximum derivatives by taking derivatives in frequency space. Note that here the error bounds are already in 3D space and there is no π/K factor involved. With these two error bounds, the resolution of the gold-standard can be determined.

4.6 Mixed-Resolution

To enable verifiable visualization with efficient hardware-accelerated trilinear filtering (instead of the ideal *sinc* filter or higher-order filters), we need to keep the data at a higher resolution. As argued above, our goal is to preserve the local maxima/minima of the reconstruction, because the trilinear filter cannot interpolate values beyond these limits. In our mixed-resolution building stage, given a set tolerance (the verification or certification stamp) some cells may be classified as subdivision cells. As mentioned above, these cells contain fine details and must be represented by additional data sample points, generated within a progressive refinement process. We store these cells separately as a progressive refinement structure.

Compared to the gold-standard the coarse volume can usually represent the data fairly well and only needs a few locations to refine. The refinement regions contain high frequencies in forms of sharp edges. The high frequencies give rise to a line-shape which is typically sparse in nature. It often occurs across the whole volume which makes brick-boundaries inefficient.

Our mixed-resolution based adaptive refinement has the storage cost of a 3D index volume. The index granularity can be chosen as a certain level of the octree. Each level has a 1/8 smaller storage (1/2 sampling rate in 1D) than its next-lower level. The error determining the local sampling rate is described as:

$$E(x,y,z) = \left| f(x,y,z) - \sum_{i,j,k \in N} w_{i,j,k}\, g(x_i, y_i, z_i) \right| \tag{26}$$

Here, the oversampled reconstruction data $f(x,y,z)$ is the gold standard, $w_{i,j,k}$ represent the filter's weights (we use the trilinear filter) and $g(x_i, y_i, z_i)$ are the grid samples at the coarser level. If the reconstructed signal error compared to the gold standard is larger than the (verified) fidelity threshold ε, we subdivide the current cell and increase the sampling rate by two.

Cell-based and node-based methods are two applicable schemes to represent octree subdivisions. We chose to implement a cell-based octree because the duplicated value on the boundary can overcome the boundary discontinuity problem well. We need the

boundary values to preserve the thin structures that often occur in medical datasets. In this cell-based method, the base cell has 2×2×2 elements, while the refinement cell could have 3×3×3 or 5×5×5 elements. A 2×2×2 base cell can be subdivided into eight children (a 3×3×3 cell) or sixty-four children (a 5×5×5 cell). Our scheme first estimates the error for a 2×2×2 cell, which is the traditionally used resolution in CT. If in this cell all the errors (compared to the 9×9×9 gold standard) are below the threshold ε, then we stop and return the refinement index as 0. If there is any estimator reporting an error larger than ε, we try a 3×3×3 cell. When the 3×3×3 cell is good then we return a positive index. Otherwise, when the 3×3×3 cell fails then we try a 5×5×5 cell whose refinement index is always returned as a negative number. We end up with a coarse volume, an index volume and two volumes.

The error checking process can be straightforwardly computed on the GPU using 3D textures mapping. The equation to guide the texture coordinate transform is:

$$v_2.xyz = (v_1.xyz - \frac{(0.5,0.5,0.5)}{l_1}) \frac{l_1 \cdot (l_2 - 1)}{(l_1 - 1) \cdot l_2} + \frac{(0.5,0.5,0.5)}{l_2} \tag{27}$$

where $v_1.xyz$ is the texture coordinate in a l_1^3 cube and $v_2.xyz$ is the texture coordinate in a l_2^3 cube. In our implementation, to better utilize the GPU bandwidth, we process multiple cells simultaneously in multiple threads, which can greatly accelerate this pre-processing procedure.

Our mixed resolution representation which has a different goal than octree-based multi-resolution frameworks, such as [25] [48] [52]. For example, the Gigavoxel framework [16] can provide interactive rendering of massive volume data. Aimed at providing aliasing-free level-of-detail (LOD), most of these multi-resolution approaches store multi-resolution data in different textures and render each brick individually. The high resolution of their input also hides, to some extent, aliasing issues [16]. Similarly, in the physics simulation domain, AMR is a refinement structure which was first developed by Kähler et al. [37] [38] and further improved by Marchesin et al. [58]. In contrast to these methods, our representation data is directly derived from the raw projection data and can so extend resolution only when needed to preserve detail. Therefore, our mixed-

resolution method is more adaptive and less brute force when it comes to data storage. In addition, we do not aim for a multi-resolution representation that can provide smooth transitions from low-resolution to high-resolution. Rather, we only keep the leaves of the octree, at the local level that preserves the fidelity of the (transformed) raw data.

T-intersections occur when two boundaries of different resolutions meet. They can cause visible banding artifacts if not handled properly. The approach by Ljung et al. [53] smoothly interpolates between these mixed-resolution boundaries, and Beyer et al. [5] introduce a scheme that blends samples nearby. This approach does not give explicit error control and therefore it is not verified by the raw data. Also, because the refinement cell position is grouped into a coarse granular octree (only 2 levels), it is inefficient to handle sparse refinement regions (void region and thin structure). In contrast, our data method can directly account for visualization errors and support finer granularity.

4.7 Continuous Boundary

In the end, there are two textures storing all refinement cells, first level and second level. The element sizes of these two volumes are 3×3×3 and 5×5×5. Along with the 2×2×2 cell in the coarse resolution, we have a 3-level mixed-resolution data representation.

Problems in this mixed-resolution representation arise when we are trying to reconstruct values on a mixed-level cell boundary. The trilinear filter itself can preserve C^0 continuity across a uniform grid. However, if we have a high resolution cell on one side and low resolution cell on the other, the interpolation scheme would result in a C^0 discontinuity. To keep with our verified rendering approach, we need to avoid schemes that blur this T-junction. Instead, we augment the T-junction with the underlying continuous data and upgrade the low-resolution side into a finer resolution. Any cell with high resolution neighboring this cell is affected by it and changes its interface accordingly. Filling the low resolution cell with original data selectively will not dramatically increase the data storage. Most importantly, feeding the gold-standard data will preserve the verifiable threshold in Equation (26).

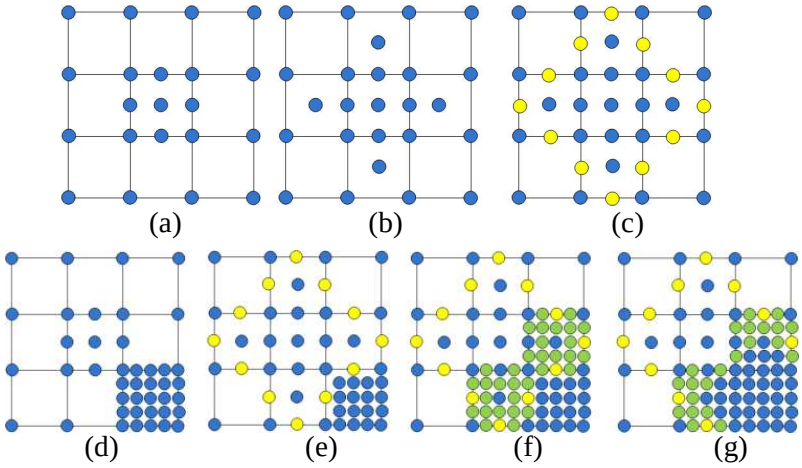


Figure 33. The data feeding algorithm ensuring a trillinearly interpolated C0 continuous boundary. A 2 level boundary region is shown in (a-c) and a 3 level boundary region is shown in (d-g).

The key idea is to only upgrade the high-low resolution boundaries, while the rest of the points are generated by trilinear interpolation (to comply with the low-low resolution boundary). Otherwise the high-low resolution boundary will propagate and we will end up with high resolution everywhere. We illustrate our data feeding process in 2D in Figure 33. In panel (a), there are eight base 2×2×2 cells and one 3×3×3 cell with refinement. Each of the 4 neighbors of the refinement cell contains a high-low resolution boundary. In panel (b), we feed the center elements from the gold-standard into the affected cells (with the same blue color). Finally in panel (c), the yellow points are further added into the 4 affected cells. These yellow points are obtained from interpolations and all the T-junctions can be removed. By adding the raw-data rather than blending, our method can adhere to previous error thresholds, and even further increase the authenticity.

The algorithm ensuring a C^0 continuous boundary in 3-level overlapping regions is shown Figure 33(d-g). If the cell is only affected by a 3×3×3 region, then data filling occurs similarly as before. If the cell is also affected by a 5×5×5 region, we first fill 3×3×3 data accordingly (panel (e)), perform another linear interpolation (green samples in panel (f)) and then fill 5×5×5 data accordingly (panel (g)).

Our method performs two processes after building the initial index of data refinement. They are the affected neighbor checking and the data feeding as shown in

Figure 33. The first process scans the initial index and generates maps recording all affected neighbors and the affected boundaries. We use 18 bits to encode different cases because between two neighboring cells, the possible sharing boundaries can be 6 faces and 12 edges resulting in a total of 18 boundaries. Thus we store 18 bits information for each 2×2×2 base cell. Totally there are totally 18×2=36 bits recording the affected cells by 3×3×3 and 5×5×5 cells.

The next procedure is filling the low-resolution cell accordingly, as shown in Figure 33. The previously generated high-resolution data are stored on disk so we do not need to re-generate the original data again. Firstly we look at the cells affected by 3×3×3 cell, fill the low-resolution cell accordingly and add those newly upgraded cells into the textures. Then we process the cells affected by 5×5×5 cell similar. The added interpolated data of a new 5×5×5 cell can be based on interpolating a 2×2×2 cell or a 3×3×3 cell.

4.8 Implementation and Results

While our method can be generally applied to high order filters, we choose the trilinear filter in this work for its efficiency, as discussed above. Our fine granular octree method lends itself quite well to GPU acceleration. We store the base volume and index volume into 3D textures. During ray tracing, each sample position is interpolated and if the current region indicates a finer subdivision then the index points to corresponding children in the octree.

To correctly fetch the index, the index volume should be shifted by (0.5,0.5,0.5)/volume_size before applying the nearest-neighbor filter. When the ray enters into a refinement cell, the local coordinates will be changed. The texture coordinate mapping between two levels follows the same procedure as the data refinement. So, if one cell has more detail and needs finer resolution, then via the index pointer, the fragment program goes to that position in the corresponding level refinement texture to fetch the data. The positive index will be directed to the first level refinement and the negative index will be directed to the second level refinement.

Finally, since our method serves as an extension of the DVR pipeline, many existing acceleration techniques available to DVR can be incorporated into our pipeline without

much modification. We currently use block-based empty space skipping and early fragment kill [23] in our visualization pipeline. But facilitated by our mix-resolution data representation, we can also readily perform ray-tracing with adaptive step sizes controlled by the local resolution.

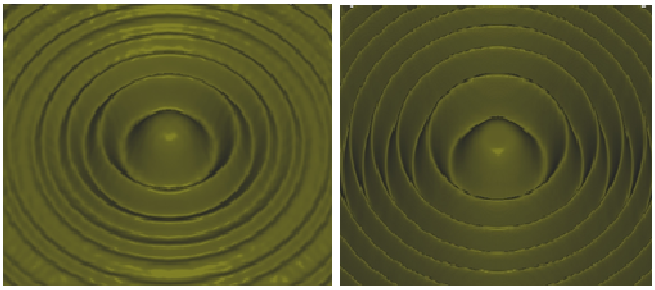
We use the FFTW library [31] to perform the Fourier and inverse Fourier transform. The GPU components are based on NVIDIA CG.

There are some implementation issues related to storage and speed. To avoid having to store all high-resolution data in memory at once, we use a block-marching approach which keeps the size of active memory reasonable. The block size should not be too small because the CT reconstruction is done on the GPU [94] (using projective textures) and the bandwidth between GPU memory and CPU memory is relatively low. Therefore we take 32×32×32 of 9×9×9 cells as a block and reconstruct all of them together. Each block is a 257×257×257 floating point data array which takes 64.75 MB of storage. This structure is scalable to large datasets with high resolution. Afterwards, some parameters in our method can also be fine-tuned for better performance. Note that the finer granularities of refinement cells will have more duplicated boundaries. In order to have better storage, we chose the granularity of the refinement to be a factor of 4. Therefore, 64 neighboring 2×2×2 cells correspond to one element in the index volume instead of 64 elements. The merged index volume then consumes about 1/64 of the storage of a traditional scalar volume.

In practice the height of the refinement component could be larger than the current dimension limit of 3D textures. It is often necessary to regroup the two refinement component. Under the 4-granular index, the refinement cell would be 9×9×9 and 17×17×17. Instead of a 9×9×9×h 3D volume texture, we can regroup the data into 90×90×9h/100. For a 17×17×17×h volume, we can regroup the data into 170×170×17×h/100. The rearrangement of the will affect the transform parameters.

We compare results obtained with traditional resolution, D^2VR, and using our verifiable pipeline. We did an experiment compares traditional volume upsampling (increasing a volume's resolution via frequency space upsampling) and our verifiable

volume oversampling (reconstructing the volume at a higher resolution from upsampled projection data). For this, we simulated a set of 74 X-ray projections (size 65^2) of the analytical Marschner-Lobb (ML) function [58]. Figure 34(a) shows the results of upsampling pipeline: from the projection data, reconstruct a volume at the base resolution of 64^3 and then double the resolution to 128^3 using frequency-space upsampling (note that this is different from sampling the original 3D ML function and upsampling it, our experiment more closely simulates a practical case – one in which an object is acquired via CT scanning). Conversely, Figure 34(b) shows the result of our verifiable pipeline: upsample the projection data by 8× using the frequency space method and then reconstruct a mixed-resolution volume (2% error threshold) that also has about 128^3 data points. We can clearly observe that both renderings are of fairly high quality, but only the verifiable method can represent all function detail (the deep grooves between the rings).



(a) (b)

Figure 34. Iso-surface rendering (using trilinear interpolation) of the ML dataset represented by the same number of volume samples: (a) volume resolution doubled via frequency space upsampling; (b) mixed-resolution reconstructed from upsampled projections and a 2% error threshold.

Figure 35 plots an error map of the results obtained with (a) D^2VR, (b) DVR (uniform resolution), and (c) VDVR (mixed resolution). For the projection data, both (b) and (c) use frequency space projection upsampling while (a) uses bilinear interpolation as described in [69]. We observe that all pixels in the VDVR panel (c) fall below the set error level of 3%, while without the verifiable mixed-resolution the error increases to 4%. D^2VR has strong ringing errors in the high frequencies (up to 6%). These error rings result in some missing or reduced high frequency rings in the function (rings 4 and 6 in Figure 36).

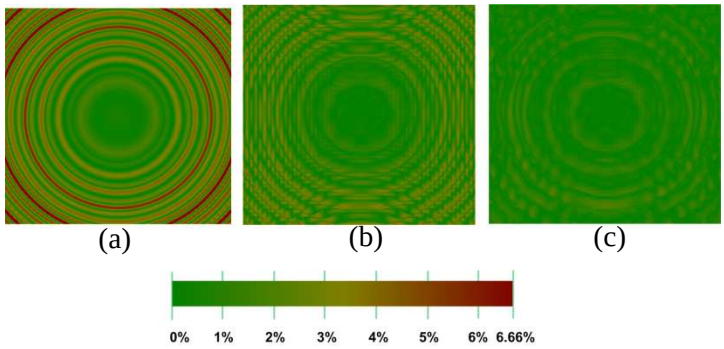


Figure 35. Error in (a) D2VR, (b) DVR at uniform coarse resolution and (c) VDVR using the ML dataset at a 3% error threshold. Projections of (b)(c) are interpolated via the frequency domain interpolation method. (b)(c) use the trilinear filter to interpolate the volume samples.

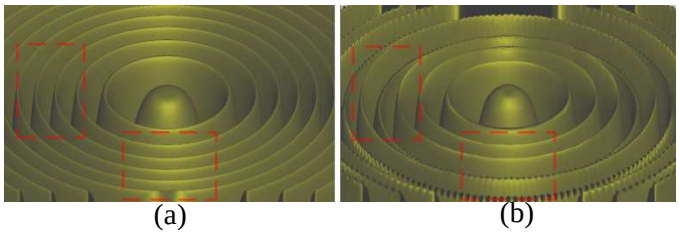


Figure 36. ML comparison between analytical function in (a) and D^2VR in (b).

We also performed these comparisons using a practical dataset. In clinical or industrial settings the CT scanner X-ray detectors often have higher resolution, and typically a bin decimation procedure I used to remove noise and aliasing. In order to mimic a real-life CT scanning scenario, we performed an 8×8 down-sampling for decimation. Figure 37 shows volume renderings of the carp dataset, where 142 projections of resolution 256×129 each were used for reconstruction to obtain both the traditional volume dataset pictured on the top and the verifiable representation pictured at the bottom. However, the trilinear filter in the X-ray projection simulation would result in strong aliasing, especially along the z-axis. The D^2VR paper [69] suggests a solution employing a spiral CT simulator. We chose a different route. We first generated an 8× high-resolution sinogram, added Gaussian noise with density 0.05, and then convolved the projections with a 5×5 median kernel and a Gaussian kernel with $\sigma = (0.1, 4.0)$ before performing an 8× downsampling. This procedure effectively eliminated the aliasing along the z-axis and also reduced the noise.

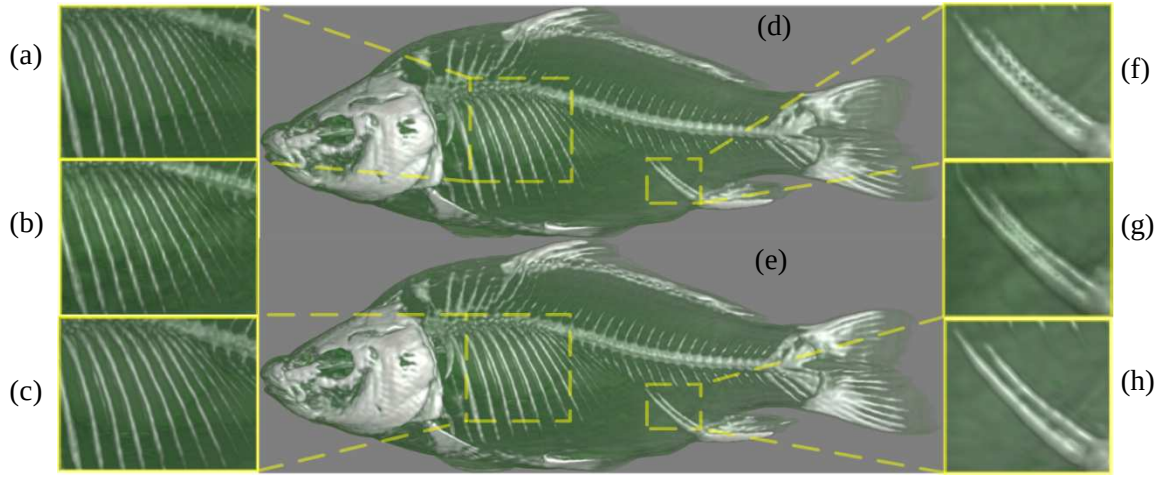


Figure 37. Renderings of the carp dataset represented in the various grid resolution types. (a)(d)(f) uniform (coarse) resolution, (c)(e)(h) mixed-resolution, (b)(d) frequency-domain upsampled resolution using the same storage than the mixed resolution (magnified cuts only).

Using these simulated projections the dataset was reconstructed and then rendered using a standard trilinear interpolation filter. Figure 37 panels (a), (d) and (f), were rendered from a uniform $128^2 \times 256$ volume (base) resolution, which is the resolution one would typically pick given the 256×129 projection data. We observe strong aliasing artifacts in these renderings. On the other hand, the renderings obtained from the mixed-resolution volume (3% error threshold, 2.4× more storage) and shown in panels (c), (e) and (h) can resolve small detail, such as the thin bones, rather well. We also took the base resolution volume of (d) and used frequency domain upsampling to generate a volume of the same storage than the mixed resolution of (e). Panels (b) and (g) show magnified cuts of a rendering of this volume. These images exhibit while aliasing is suppressed, small detail cannot be solved. The highlighted fish tail is shown in Figure 29 which compares our VDVR with D^2VR. Panel (a) shows that D^2VR still smoothes out some fine details on the tail, while VDVR in panel (c) provides sharper images, showing the thin bones clearly.

The performance was measured on a PC with Intel Core 2 Duo 3.00GHz CPU, 1.75GB RAM, and a NVIDIA GeForce 9800 GX2 GPU. Table 2 gives insight into the rendering speed and compares it with D^2VR. For these results, we rendered into a 512×512 window and used a ray step size of 1. Volume textures were RGBA 32 bit

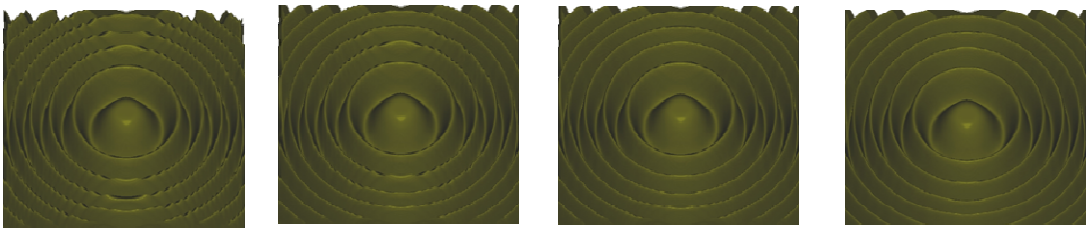
floating point. Central-difference-based gradient estimation was used in the volume rendering. Since the Shepp-Logan dataset has very low contrast tumors and high intensity bones, verification was only performed on intensities around this tissue intensity range.

Table 5. The performance of VDVR with 3% error tolerance

Dataset	Base Resolution (# of 2×2×2 cells)	Pre-Processing	Rendering	Storage
Shepp-Logan	128×128×128	12 min	10 fps	4×
Carp	128×128×256	35 min	11 fps	2×
Beetle	102×62×128	6 min	13 fps	5×
ML	64×64×64	4 min	20 fps	6×

In Table 5, the error threshold was set to 0.03 for all datasets. We observe that a rendering speed of about 10 frames/s is possible for practical datasets. VDVR requires about 4 times more storage than DVR and likewise D^2VR. However, we also note that if D^2VR were to store high-resolution projection data to overcome the aliasing effects incurred from sampling ramp-filtered projection data with an inferior filter, its storage requirements would be significantly higher.

Figure 38 shows the qualitative effects of the error threshold, using the ML projection data. It appears that refinements will first pick up high-frequency details (the outer rings), and then expand to lower-frequency details (the ring close to the center). Eventually, fine details no longer improve due to the implicit band limit of the ML.



4% 3% 2% 1%

Figure 38. ML rendered with VDVR at different error thresholds.

Finally, Table 6 examines the effect of different error thresholds on rendering performance. While the ML function is small, it is rich in fine details. Therefore its storage increases dramatically when the error threshold is lowered. The rendering speed,

however, is stable since the required rendering effort is still small for the GPU. Conversely, the carp dataset is larger in size overall and so requires a larger rendering effort, but it contains only sparse details. Therefore the storage is less sensitive to the error threshold.

Table 6. The effects of different error thresholds

Dataset	Threshold	RMS	Rendering	Storage
ML	4%	0.0156	20 fps	2.34×
	3%	0.0123	20 fps	6.04×
	2%	0.00917	20 fps	8.35×
	1%	0.00747	20 fps	32.0×
Carp	3.0%	NA	12 fps	2.41×
	2.5%		11 fps	3.03×
	2.0%		8.6 fps	4.23×

Chapter 5. View Suggestion

In this chapter we concentrate on the problem of suggesting to the user a set of potentially good views. We leave the design of the transfer function needed to highlight the exposed features to the user, supported by suitable existing interactive algorithms. There are many of these (e.g. [14] [14]) and so we do not discuss them in this work.

We propose a viewport advisor — iView [107] — rooted in high-dimensional feature space which does not rely on transfer functions or volume segmentations as an initial input. By applying high-dimensional feature clustering, our proposed method can automatically detect salient composite features and based on this analysis suggest promising viewpoints. The user may then inspect these promising views more closely by interactively exploring the transfer function, or run one of numerous automatic algorithms to optimize visibility. We henceforth call this method viewpoint suggestion since it helps users to navigate to favorable positions that potentially show interesting structures. In this way our approach is less ambitious than a full-fledged view selection pipeline, but at the same time more versatile and appropriate for data exploration. It supports the extraction of a set of good views appropriately refined with different transfer functions instead of a set of good views limited to one fixed transfer function.

Our approach also offers advantages in terms of interactive data exploration. This is especially important when users are not certain about the properties of the structures of interest and need to refine the renderings. Traditional view selection algorithms [7] [10] involve full volume rendering from all possible viewpoints and calculate the viewpoint entropy from all voxels. This can result in prohibitive wait times if users want to update the transfer function during the search for the best viewpoint. In contrast, our method splits this process into two separate and subsequent phases: (1) the determination of a set of good viewpoints using relatively inexpensive operations (GPU-accelerated clustering, cluster-based visibility test and entropy calculation), and (2) the interactive refinement of the transfer function from these promising viewpoints to generate the desired salient volume rendered images. This approach allows users to maintain a stable spatial

reference (and one at which many features are visible), while exploring these features and their relationships one by one by modifying the transfer function.

To address the problem of fully covering all features in a volume, our viewpoints suggestion method provides a two-fold solution. First, our system not only suggests a single best viewpoint, but also provides an interactive navigation interface where all the views are color-labeled with a relative measure of feature exposure. Also, the system can progressively mark visited features and only show the distribution of the unknown, yet undiscovered information. This type of interaction has already been useful in specialized applications, for example in virtual colonoscopy [40] where colon wall patches already viewed are painted in green on the fly-through, enabling doctors to focus their attention onto unpainted areas. Second, to effectively guide users in their exploration of the entire volume, our system provides an automated viewpoint suggestion module, effectively minimizing the number of views to be inspected. Our multiple-viewpoint suggestion algorithm seeks to determine the minimal set of views that can cover all features, which maps to the Set Cover Problem (SCP). Our set-cover problem solver together with the interactive viewpoint navigation tool then aids users in gaining a complete understanding of the features in the volume.

Our view suggestion pipeline is shown in Figure 39. The first stage is a multi-dimensional data clustering. Given a certain noise-level for the dataset, we consider voxels with high gradient/normal variation as the important features. We perform k-means clustering to group voxels into blobs, followed by a visibility test. In the next stage we compute the information gain for all viewpoints around the object and create an entropy map that we display on a sphere that doubles as a track-ball interface used to change viewpoints. Thus, by mapping the entropy map directly on the track-ball, users can directly and intuitively identify and navigate to favorable view locations. The user can also add or delete viewpoints by clicking on the sphere and the displayed entropy map is updated accordingly.

Alternatively, the SCP solver can be used to suggest to the user a set of optimal or at least near-optimal viewpoints from which to visualize the volume. It provides a series of

viewpoints that covers all features. The set of optimal views suggested by the SCP solver is annotated onto the navigation sphere and the user can continue navigating through the sphere with these added suggestions in hand. In the meantime the user can also use the transfer function designer to explore settings that expose features at the given viewpoints.

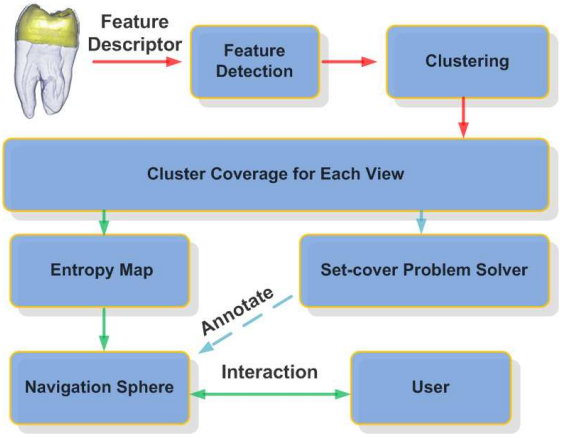


Figure 39. Viewport suggestion pipeline.

5.1 Viewing Entropy

Information theory defines entropy as a measure of uncertainty associated with an information source. Since, to resolve this uncertainty, the amount of data we need to transmit to the receiver defines the amount of information content, entropy of the source hence also measures information. Let us consider any information source A which transmits a random sequence of symbols taken from alphabet $\{a_0, a_1, \dots, a_{K-1}\}$ where occurrence probabilities are $\{p_0, p_1, \dots, p_{K-1}\}$. Entropy of this information flow is given by,

$$H(A) = -\sum_{i=1}^{K} p_i \log p_i \tag{28}$$

Now, say, in addition to the given probabilities, the receiver also has the knowledge that a certain symbol a_x is always followed by some other symbol, a_y. Presence of this knowledge to the receiver, let us define it as *E*, reduces uncertainty regarding the source. Entropy after this knowledge would be, $H(A|E)$.

In the context of volumetric data, we have an information source – the volume itself, the information receiver – the viewer and a transmission process which includes the whole pipeline of volume rendering. If the volume is not shown to the viewer, then the uncertainty associated with the volume is at maximum and it represents the total information content of the volume, say we denote this by $H(X)$. Now, in the event that we render a particular view v_1, partial information of the volumetric features become revealed to the user. The uncertainty remaining can roughly be defined as $H(X|V = v_1)$. Since we are interested in finding the view that reveals most information, from the perspective of information theory, what we want is to find a view v_{min} that minimizes $H(X|V = v_i)$ with respect to all possible views. From the chain rule of entropy we can write,

$$H(X|V = v_i) = H(X, V = v_i) - H(V = v_i) \tag{29}$$

Here, $H(X, V = v_i)$ is the information content of the volume and its view taken together. $H(V = v_i)$ denotes entropy of a particular view and as such is a measure of information content of a rendered view. Since a view is just a projection of the volume data, we can consider $H(X, V = v_i)$ to be constant across views. Hence, minimizing $H(X|V = v_i)$ effectively means maximizing $H(V = v_i)$. So, a good view is identified as the one that has large view entropy.

Vazquez et al. [86] [87] firstly applied the concept of viewpoint entropy to determine the best viewpoints for polygon-based scenes. In volume rendering, a straightforward way [7] [81] to measure the view entropy requires a transfer function to perform volume rendering for the view. Bordoloi et al. [7] proposed to use voxel-based entropy to select viewpoints in volume rendering, assuming that transfer functions are given. Takahashi et al. [81] proposed a similar framework based on iso-surface entropy, weighted by a given transfer function. Chan et al. [10] extended Bordoloi's work by considering spatial relations between structures, after a user-specified segmentation has been given. Our viewpoint suggestion algorithm is fundamentally different from these works as it does not depend on either prior transfer functions or segmentation.

Most of the recent research in view selection uses metrics based on scalar values to locate regions of interest, that is, before selecting the views the user is either required to design a 1D transfer function [7] [81] or perform a segmentation [10] to classify these regions. Then, once this has occurred, the viewpoint selection algorithm will search for the best viewpoint to display the maximum amount of information. However, there are a number of potential pitfalls with this methodology. First, before we can start searching for the best view, the user is required to know the scalar values of the hidden structures to be classified. But without having a proper look at the data first (and without the presence of strong domain knowledge), the user might not have a clear idea what these structures actually are, even in a coarse sense. Any initial guess will likely not be able to classify the hidden features successfully, and so the view selection algorithm in turn will not help to find the best viewpoint. Also, due to the fact that these methods require input in form of a transfer function or segmentation, if the user decides to change either of these a re-computation of the entire pipeline is needed to suggest the new best view. This iterative process can potentially take a long time and thus makes exploring transfer functions in an interactive manner impossible. Secondly, many structures in 3D volumetric data require more than scalar values to be classified properly. They may require gradient magnitudes (or even higher-order metrics), in conjunction with multi-dimensional transfer-functions. Given all these inherent shortcomings, purely scalar-based view selection algorithms can be quite limited for practical use.

The selection of multiple views or view planning has found application in many domains. A variety of methods seek to solve the next best view problem, such as the entropy-based method [90], the visibility-based method [30], and the silhouette-based method [1]. They have wide application in the placement of laser sensors [6] and RFID sensors [92] and for determining the best circular trajectory [2] or angles [108] in cone-beam CT. These view planning methods cannot be directly applied in volume rendering but they can provide useful insights.

5.2 Entropy Calculation

5.2.1 Feature Descriptor

Our method is based on feature detection and high-dimensional data clustering. As for the question how to define features in high-dimensional space as potentially interesting structures, there are many choices and their suitability can be application-dependent. We chose to provide a very general and application-neutral importance metric based on normal-variation. Normal-variation plays a significant role in lighting. In this work, we shall assume that an area with large normal/gradient variations contains salient information, while regions with similar gradient directions have less information. This metric is an extension of 2D curvature estimation and it generally belongs to the group of Laplacian operators. The discrete importance estimation for a voxel located at (x_0, y_0, z_0) is:

$$w(x_0, y_0, z_0) = \sum_{(x,y,z)\in N(x_0,y_0,z_0)} |\nabla f(x, y, z) - \nabla f(x_0, y_0, z_0)| \qquad (30)$$

where $f(x, y, z)$ is the volumetric scalar field, ∇ is the gradient operator and $N(x, y, z)$ is the set containing a neighborhood of (x, y, z). In our estimation, only the 6 closest neighboring points are considered. This metric is different from the classic Laplacian operator which is defined as the divergence of the gradient vector field and thus can be negative. The importance weight here sums up the absolute values of gradient difference individually which can guarantee the weight to be positive. The intuition behind this metric is that we want to have a measure of the perturbations of gradient/normal in a region.

Most practical volume data contain a certain level of noise which will affect the feature detection. In the pre-processing stage, we take the ambient noise level as an input to threshold the scalar values. We also consider the noise removal as a thresholding procedure on gradient variation. After applying these thresholds, the resulting voxels are considered the important voxels and are clustered in a five-dimensional space: scalar value, gradient magnitude, and (x, y, z) coordinate.

5.2.2 K-Means Clustering

The k-means algorithm is one of the most well-known clustering algorithms. Given an input value k, it can partition n objects into k clusters based on some similarity (distance) measure. We apply k-means clustering to get k blobs in 5D space. We also record the voxels inside each cluster and remove a cluster if the number of voxels is too small (less than 5). The gradient/normal direction for each cluster is computed as a Gaussian-weighted average which enhances spatial coherence.

Figure 40. K-means feature clustering with gradient vectors shown for (a) a standard cube and (b) a cube with text on the back surface.

An example of computed gradient vectors is shown in Figure 40. The clustering phase can employ automatic feature detection if the ambient noise level is known. We build our system in the high-dimensional feature domain. Hence it can detect local structures with high gradient variation and adjust views for these local features. This provides a general importance metric well suited for non-expert users, to minimize user invention. But we note that our data-clustering pipeline can readily support other more specific metrics if more specific domain knowledge about dataset and task is available.

An important aspect in k-means clustering is the input value k. The value of k controls the resulting number of clusters and as such the grouping of similar features of the input dataset. For a given dataset, to identify the features distinctly, the algorithm requires a certain k that will ensure separation of features such that the clusters truthfully represent the features. Having a smaller k value will merge a set of features into a big cluster, whereas a larger value will produce clusters covering fine details. From a multi-resolution point of view, the choice of k affects the resolution of the features to be extracted. Therefore, the value k will reflect the average size of the feature clusters. We base the choice of k on the average cluster size (and therefore desired level of structure

detail), chosen by the user via a slider interface. Then the value k is found by dividing the total number of noise-free voxels by the desired average cluster size. But alternative approaches such as the elbow criterion [42] and X-means [66] may also be utilized to obtain an appropriate k. Finally, users may also inspect the clusters by visualizing them in the 3D interface.

Another factor in k-means clustering is the choice of the initial k seeds, which can produce a certain level of randomness in the clusters formed. To overcome this problem we use the standard practice of clustering the data multiple times with different random seeds and picking the clustering that has the least overall L2 error with respect to the clustered data points. Since we use the GPU-based k-means library [27] the performance hit is relatively minor.

5.2.3 Transfer-Function Independent Entropy

Once we have defined the feature set, what we then need to find are the views that can show the features distinctly on the screen through graphical rendering. To facilitate comparison among all these views, we assign a score to each of them. And to compute this score, we apply concepts from information theory in a similar fashion as in previous work [7] [10] [81].

Our entropy estimation is different since we do not want to involve transfer function as an input and later restrict the decision on a single fixed transfer function. Instead, we propose to measure the maximum possible information (across all possible transfer functions) for a viewpoint. This is possible since the 2D image generated by computing the volume rendering equation depends on not only the transfer function but also on the shading (lighting) effect. Shading plays a significant role in conveying information about shape and is well-studied in computer graphics. We define potential information for shading as blobs of voxels and we compute the entropy based on how well one can resolve these feature-clusters at a given view based on the shading (lighting) effect. It serves as an extension of Bordoloi's work [7] in which the visibility of each voxel is computed together with the transfer function to evaluate the entropy. Here we group voxels in the volume according into clusters. Then probability distributions associated

with voxel-clusters are needed to calculate entropy. Let q_j represent the contribution of cluster j in a viewpoint. q_0 is a special case indicating the background volume (containing all voxels that do not belong to any clusters). Then the view entropy for a certain view $H(V = v_i)$ is:

$$H(V = v_i) = -\sum_{j=0}^{K} q_j \cdot log_2 q_j \tag{31}$$

$$q_j = q_j(V) = \frac{1}{\sigma} \cdot \frac{VC_j(v_i)}{W_j} \quad where \quad \sigma = \sum_{j=0}^{K} \frac{VC_j(v_i)}{W_j} \tag{32}$$

Here K is the total number of feature clusters. The factor σ will make sure that all q_j sum up to 1. $VC_j(v_i)$ is the visibility of cluster j in view v_i. W_j the noteworthiness [7] of cluster j, is defined as:

$$W_j = I_j = -log_2 p_j = -log_2 \frac{|n_j|}{\sum_{i=0}^{K} |n_i|} \tag{33}$$

where p_j represent cluster probability, calculated from the number of voxels in each cluster normalized by total number of voxels. We add the consideration of background. n_0 is number of voxels that do not belong to any cluster, p_0 is the probability of background and W_0 is the noteworthiness factor for background.

For the cluster-based entropy, we mark all voxels that do not belong to any clusters as ‘background’, which is similar to the background feature definition in viewpoint selection methods for polygonal models [86] [87]. This will remove singularities where only one cluster is shown in a viewpoint but its entropy is 0. After considering all background voxels, if no feature cluster is shown, the view will have zero entropy. In contrast, if any feature cluster is shown then the entropy will be non-zero.

When we are suggesting views onto the volume, besides finding a single view that reveals information in the best way, we also want to guide the user such that they will not miss out on any of the features. In the language of information theory, a single view v_i might be the optimal view in terms of view entropy but it may be the case that $H(X|V = v_i) \neq 0$. So, we propose to suggest to the user a set of views $\{v_1, v_2, \dots v_j\}$ so

that $H(X|V = v_1, V = v_2, .., V = v_j\}$=0, that is, find a set of views that collectively can give the viewer a total picture of all the important features of the volume data. Chapter 6.3 shows how we compute such set by solving the set cover problem.

The k-means algorithm outputs a series of 5D clusters and gradient/normal values at the centers of the clusters. For each cluster, we extract its spatial information as a 3D ellipsoid and estimate visibility based on the cluster's normal direction.

We assume the center of a cluster to fall within the clipping window. Then there are three major factors that contribute to a good view: (1) the angle between the cluster's normal and the viewing direction (the eye ray), (2) the number of clusters that can be shown, and (3) the total number of voxels within potentially visible clusters.

Our goal is to calculate the maximum potential entropy for the feature clusters. We established a set of criteria for a feature cluster to be classified as invisible. We first set a threshold on a clusters' normal range and later use entropy to measure viewpoint quality.

The first criterion is that the gradient direction and eye-ray should be within a certain range, enabling shading effects to enhance small detail in the volume rendering. Shading conveys a strong cue for shape (see "shape from shading" in computer vision, graphics and robotics [105]). An important observation is that at 45° the Lambertian cosine shading functions starts to loose strength, since the derivative of the cosine function has an extreme point in the vicinity of 45°. So if the normal vector of the cluster and viewing vector make an angle of greater than 45°, our method rates the viewpoint as inadequate to cover the feature cluster's information.

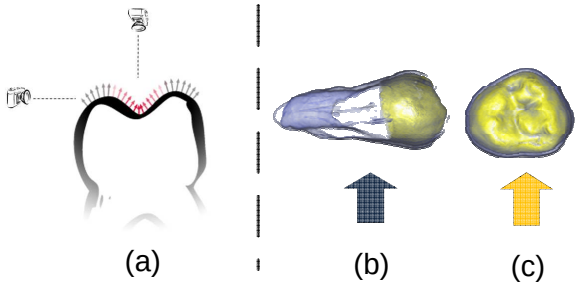


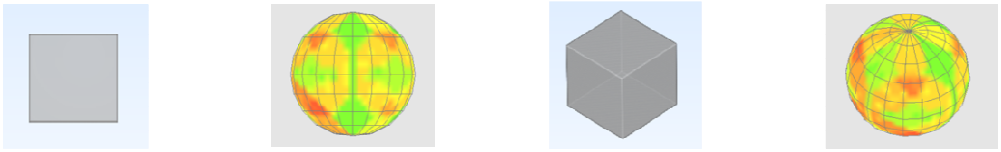
Figure 41. Silhouettes fail to convey concave shape.

Silhouettes can also be salient in conveying shape information (this is a popular method in non-photorealistic rendering). Silhouettes start to become visible when the view and normal vectors are close to 90°. Consequently, we may allow clusters that are close to 90° to be visible as well. This criterion is especially effective in dynamic flow visualization, in which interesting wave fronts can be represented as silhouettes. But in static data with concave shapes, it may produce poor results as shown in a simple example (Figure 41). Figure 41(b) is a good view via the silhouette criterion but it provides only little information. In contrast, Figure 41(c) emphasizes only normal deviations and is a more informative view. Hence, we find that shading effects tend to be a safer way to test the visibility of the features, especially when the user is facing non-convex shapes. Since in this work we only focus on volume rendering of static datasets, we prefer the shading-based method to point out salient details.

Our visibility test so far did not account for the occlusion among clusters, since our target was the maximum possible exposure of all details within a given viewpoint. This is less of an issue since advanced methods (such as occlusion-spectrum based transfer function) have the capability to explore data with occlusions. In addition, conceptually we place no limitation on the number of images or transfer functions per viewpoint. As such, at a given viewpoint, the user may theoretically see all the structures within a normal range by applying different transfer functions one by one. In fact, this was our initial design choice: giving the user the freedom to choose any type of transfer function or take any number of rendering results later on. But practically speaking, the occlusion effect among the ellipsoids represents an additional time overhead (and therefore cost) in the data exploration process. So we provide a weight by which the user can set a preference on less occlusion which in turn eases the transfer function design.

By applying the visibility test, we measure the quality of a viewpoint in terms of view entropy (Equation (34) and (35)), and find the largest entropy by extensive search. As discussed, the major difference between ours and other work [7] [10] is that instead of representing information according to scalar value, we define it on important features in a high-dimensional feature descriptor domain.

5.3 Viewpoint Information Exploration



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

Figure 42. Viewpoint navigation for a cube dataset. (a): display window for the rendered object. (b): view navigation sphere (track-ball). Upon rotating the track-ball. Both (c) the rendered object and (d) the view navigation sphere will rotate in a synchronized manner.

If we assume that all viewpoints have the same camera distance, the possible projection locations will be on a sphere with radius R and can be parameterized by longitude and latitude. We further assume that in each view position we use the same field-of-view (FOV) and we are looking at the center of the sphere. Rendering the viewpoint entropy map onto a sphere, every single point on this sphere then represents one view position and the intensity of a point denotes the amount of information this viewpoint can possibly cover. As shown in Figure 42, the nearest point to the user (screen center) is the current viewpoint. The navigation window and volume rendering are displayed side by side. The user can rotate the sphere in the arc-ball interface and in the meantime the volume will rotate in a synchronous fashion. The user can then check on the map which view position would possibly be more interesting to look at.

Deviating from the previous works that use single viewpoint selection interface, we provide progressive navigation tools. Users can select a complete set of views to render the volume data in a greedy manner. The system helps the user to navigate and allows them to select a series of viewpoints. When the user marks a point on the sphere to represent the selection of a viewpoint, the system shows the rendered image and updates the entropy information interactively, reducing the entropy map to that of the undiscovered features. Users may then continue to choose several more views until not much color is left. Also, the user may undo the latest selection and the system will then add the affected clusters back into the map. The user may also undo selections multiple times which will be reflected on the entropy map in reverse order.

We then explain how to update the entropy calculation after a user's interaction. Initially, all clusters are set as unknown, denoted as $u_j = 1$ for the jth cluster. The clusters that have been explored by the user are marketed as inactive. Thus:

$$u_j = \begin{cases} 0 & \text{if user visted cluster } j \text{ and } j \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{34}$$

$$q_j = u_i \cdot \frac{1}{\sigma} \cdot \frac{VC_j(V)}{W_j} \tag{35}$$

Undoing a user selection will mark the corresponding u_i as active again. We do not perform further normalization of the noteworthiness factors during user selection, considering the fact that the amount of total information is constant. In this way, the user will observe the color fading from red to blue to convey the amount of information that has now been explored.

As most users tend to use this tool in a greedy manner, it may not be the optimal way for choosing the camera positions that cover all detected features. For this purpose, we incorporate our SCP solver to help users to find the optimal viewpoints. This is explained next.

5.4 Ant Colony Algorithm for Set-Cover Problem

We suggest an optimal series of viewpoints by solving the SCP. We first generate a large number of views as candidates to choose from. Each view will then cover a number of features. Thus we make each view a *set*, while the features that need to be covered form *elements*. The optimization objective is to find the minimum number of views that cover all salient features.

The SCP was one of Karp's 21 NP-complete problems [39]. A mathematical model for the SCP is usually described by a 0-1 matrix. S. Let $A(a_{ij})$ be an m-row, n-column, zero–one matrix. We say that a column j covers a row i if $a_{ij} = 1$. Each column j is associated with a nonnegative real cost c_j. Let $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$ be the row set and column set, respectively. The SCP can be stated as:

$$\min\{\sum_{j=1}^{n} c_j \cdot x_j\} \tag{36}$$

subject to

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \geq 1 \text{ and } x_j \in \{0,1\}, \qquad \forall i \in I, \forall j \in J \tag{37}$$

where, $x_j = 1$ if set j is selected, otherwise $x_j = 0$. The matrix $A(a_{ij})$ encodes the all viewpoints strength to cover the feature clusters in the volume.

The SCP can be solved by many algorithms and the ant colony algorithm is one of the fastest solvers [68] [70]. It is inspired by the observation of real ant colonies. The general mind-set behind ant colony algorithm is that a large amount of artificial ants search for an optimal solution defined by Equation (39). Each artificial ant chooses a set one by one until it achieves a complete cover defined by Equation (40). The decisions for choosing different sets are partially based on Russian-roulette. Additionally, the probability for choosing one set will increase if a large number of ants choose it, which is in the way of pheromone information exchange. In the following, we explain the ant colony algorithm for the SCP in detail.

The probability for an ant to choose set j is based on the state transition rule:

$$P(s_t = j|S_{t-1}) = \begin{cases} \dfrac{\tau_j h_j^{\beta}}{\sum_{q \in J \setminus S_{t-1}} \tau_q h_q^{\beta}} & \text{if } j \in J \setminus S_{t-1} \\ 0 & \text{otherwise} \end{cases} \tag{38}$$

where S_{t-1} denotes the partial solution constructed before step t, $j \backslash S_{t-1}$ denotes the subset of unselected columns, s_t denotes the set that will be chosen at the step t and the parameter β ($\beta >= 0$) determines the relative importance of the heuristic factor with respect to the pheromone.

The heuristic is usually defined by a greedy method. If R is the set of still uncovered elements and c_j is the cost associated to set j. The heuristic value of set j is:

$$h_j = |I_j \cap R| / c_j \tag{39}$$

The pheromone trials are stored at each set as τ_j. They are initially set to zero and updated later as:

$$\Delta\tau_j = \begin{cases} \frac{1}{\sum_{q \in S_{gb}} c_q} & j \in S_{gb} \\ 0 & otherwise \end{cases} \quad (40)$$

$$\tau_j \leftarrow (1-\rho)\tau_j + \Delta\tau_j \qquad \forall j \in J \quad (41)$$

where S_{gb} is the current best solution across all ants, ρ is pheromone evaporation factor (with $0 < \rho < 1$) and $\Delta\tau_j$ is the amount of pheromone put on column j. The range of pheromone should be clamped with a range $[\tau_{min}, \tau_{max}]$ where:

$$\tau_{max} = \frac{1}{(1-\rho)\sum_{j \in S_{gb}} c_j} \quad (42)$$

$$\tau_{min} = \varepsilon \cdot \tau_{max} \qquad 0 < \varepsilon < 1 \quad (43)$$

For more a detailed description of the ant colony algorithm applied for the set covering problem, the reader is referred to [70].

5.5 Suggesting Best Combination of Views

As mentioned, a greedy search is not always the most optimal approach when searching for a set of views covering all clusters. The entropy-rated viewpoints are only a result of a local heuristic which is not adequate to find a global optimum. Hence we provide the user with the minimum number of viewpoints needed for full exploration based on an ant colony optimization of the set covering problem.

The ant colony optimization method creates *n* artificial ants to search for the viable solution to Equation (39). After all *n* ants find viable solutions the system keeps track of the best ant (with minimum cost), updates the pheromone and runs n more artificial ants until the desirable cost is found by an ant. In our problem, the user can specify the number of views he/she wants to have to expose all feature clusters. Then the SCP solver will run multiple ants to search for the solution. Each ant will make a decision on what is the next viewpoint to choose based on probability and pheromone. The probability is proportional to the number of unknown clusters that can be covered and the pheromone. And the pheromone reflects how many other ants previously chose this viewpoint. After

all ants have finished, the ant with the minimum number of viewpoints will deposit the pheromone to the viewpoints it chose. The system will report a success if an ant has found the desired solution. If in a limited amount of time, the ants cannot find a set of views under the desired cost, the system will report that no solution has been found.

We have implemented the ant colony algorithm in C++ and validated it against several test problem cases [4] with known solutions. The ant colony algorithm can run for a given limit of time and the time limit for search is set to 10 seconds in our method.

5.6 Results

In our experiments, we set the viewing angle limit to 45° and assuming a voxel distance of 1 mm we set the viewing distance to 5,000 mm. The image size is always 512^2 pixels. The view positions on a sphere are sampled on a 60×30 grid, with a total of 1,800 viewpoints. All volume-rendered results shown in this work were obtained using the rendering software Imagevis3D [27] [32].

We first tested our method on a simple cube dataset. The cube's size is 80^3, residing in a 256^3 volume grid. We added a shift vector (10, 20, 30) to the cube which moves it off the volume grid center. Figure 43 displays the rendering results obtained with our system. In this case, the SCP solver automatically suggests 4 different views, looking down the cube diagonals. All of these 4 views coincide with the best views provided by Bordoloi et al. [7]. It appears that two images are not sufficient since each viewpoint will resolve three edges in the center through shading, while the rest of the edge features are partially but not completely visible. As seen in Figure 41(c), it is not safe to only have two views to visualize the cube, since we do not have good information about the 6 edges appearing in the silhouettes. Conservatively speaking, only 4 viewpoints will be able to see all 12 edges with full exposure of all features. The corresponding entropy map is shown in the bottom row of Figure 43. The initial high entropy map with no views selected shows 8 favorable regions, identifying the cube's 8 vertices as best views. The entropy map is then updated gradually as views are selected. Here selecting a given view typically removes more than one local maximum from the entropy map.

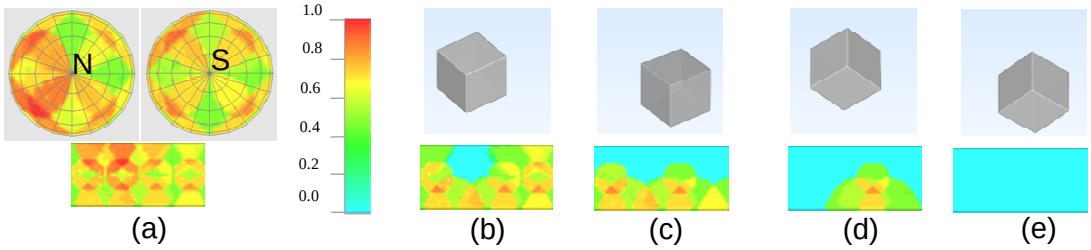


Figure 43. standard cube dataset (4 viewpoints computed by the SCP solver). The cube is shifted from the center. (a): the initial entropy map. (b-e): the suggested viewpoints rendered (top) and the maps with remaining entropy (bottom).

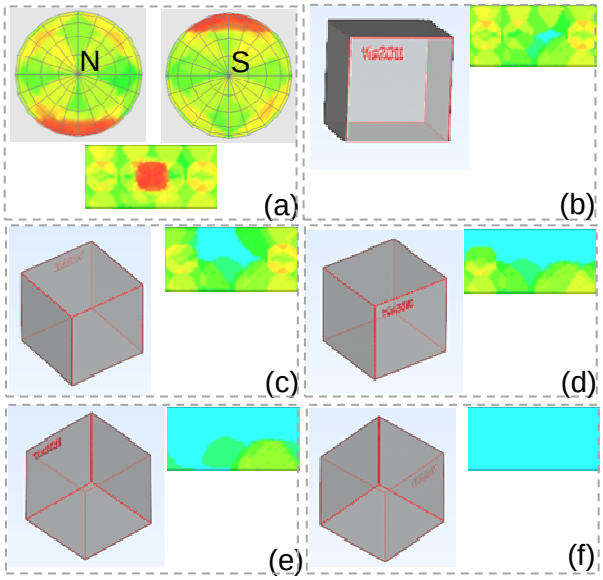


Figure 44. The cube with text on one face (5 viewpoints computed by the SCP solver), with transfer function highlighting the text. (a): the initial entropy map. (b-f): the suggested viewpoints rendered (left) and the maps with remaining entropy (right).

For our next experiment, we add the text “Vis2011” on one of the surfaces of the cube (normal perturbation). The results for this modified cube are shown in Figure 44. In this case, we need 5 views to fully visualize the dataset. In navigation mode, the entropy map clearly highlights the surface with text, indicating that in this region there is something important. In the automatic view suggestion mode, one of the resulting views specifically targets the text while the other viewpoints aim to look along the diagonals. In contrast, scalar-value based methods [7] [81] will not be able consider the text as an important feature and so will display the entropy map of a uniform cube. Our method, on

the other hand, can faithfully detect this type of intricate surface feature and suggest something of possible interest is hiding in a certain view.

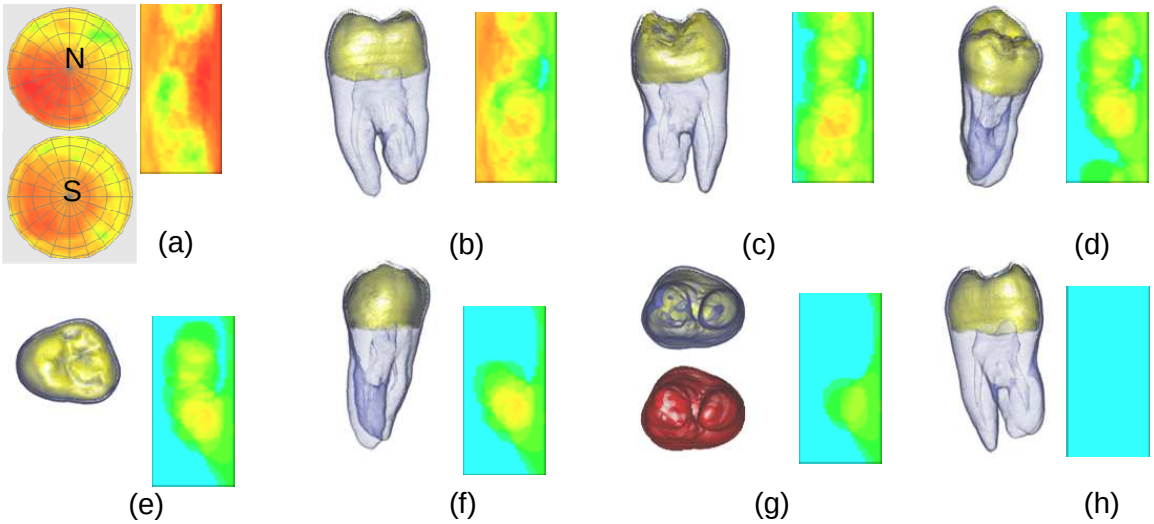


Figure 45. The tooth dataset – the set of 7 salient representative viewpoints returned by the SCP solver. (a): the initial entropy map, (b-h): the images rendered from the suggested viewpoints (left) with remaining entropy map (right). (g): modifying the transfer function to see the detailed shape of the tooth surface.

We also tested our pipeline on medical data, which typically do not have strong regular edges like the cube. Our experiment uses the well-known tooth dataset. Figure 45 shows the resulting 7 views suggested by the SCP solver (the progressive entropy maps are shown on the right of each image). We note that both the rendering results and the entropy maps have been re-oriented using the user-defined up-directions since the default up-direction does not conform to the user-preferred tooth direction. In accordance with Bordoloi's result, the entropy map of this dataset indicates the north-pole and the south-pole as the two most interesting regions. The SCP solver subsequently chooses the north-pole and the south-pole as the first two viewpoints, and therefore the resulting 2 images are deemed to reveal the most interesting information on the data. We also see that after the first 2-3 views have been selected the remaining entropy is rather low and sparse, but our system includes them in the gallery to provide complete coverage. Another important aspect to note in this particular experiment is the utilization of multiple transfer function in the same view. For some of the viewpoints, the potential information is hidden if only one transfer function is considered, as shown in the 6th viewpoint in Figure 45(g). But

once the user is given a view that can guarantee the presence of useful information, he can always refine the transfer function to freely explore the interesting structures best revealed by this view.

Next, we test our method on a practical dataset obtained from CT scanner. Figure 46 shows the results for the carp dataset. According to the entropy maps, the first 3 views cover the most important features of the carp. The map also indicates that one side of the carp is slightly more interesting than the other side, due to the carp's bent body as shown in Figure 46(d). The remaining two views are less important, but again we provide them for completeness of the gallery.

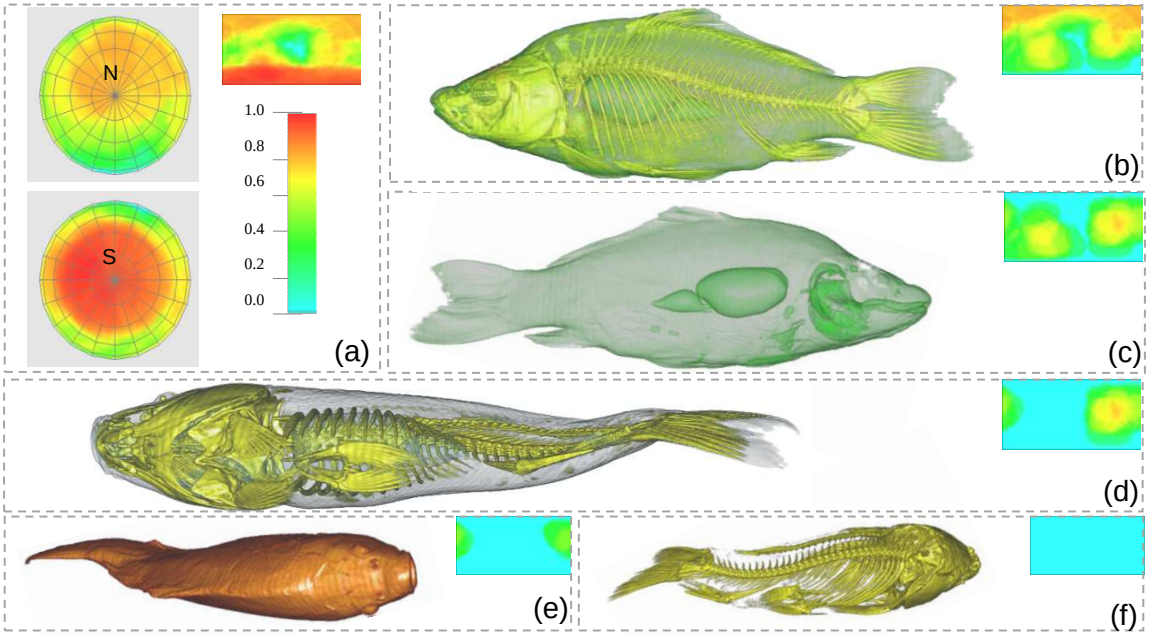


Figure 46. The carp dataset (5 viewpoints computed by the SCP solver). (a): the initial entropy map. (b-f): the suggested viewpoints rendered (left) and the maps with remaining entropy (right). The transfer function could be changed in different views.

Figure 47 shows a gallery obtained for the engine dataset where panels (d) and (g) each show images of the same view but rendered with different transfer function settings to visualize different aspects of the engine. This study illustrates the benefit of the two-phase design of our system. First obtain a view direction at which many different types of features can be observed well, and then maintain that view and visualize these features in a number of ways to accentuate their various relationships in turn. This approach allows

the user to maintain a coherent spatial reference, while learning about the dataset through dynamic feature exploration.

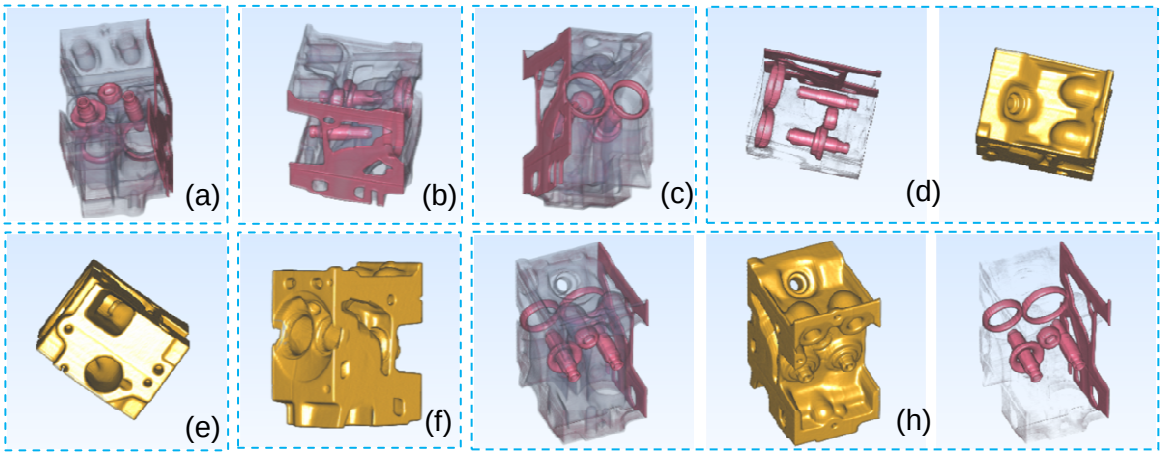


Figure 47. The engine dataset (7 viewpoints computed by the SCP solver). (a-g): the suggested viewpoints rendered with different transfer functions. (d) and (g) shows the need to use multiple transfer functions to explore features.

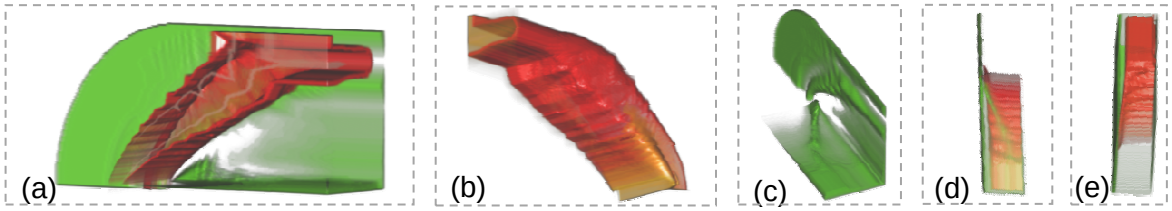


Figure 48. The bluntfin dataset (5 viewpoints resulting from the SCP solver).

Finally, we also tested our method on a fluid simulation dataset, e.g., the well-known blunt fin. Figure 48 shows the 5 views suggested by the SCP solver. The view in Figure 48(a) is suggested as the most important view, which in fact is close to the most commonly selected view onto this dataset.

Our experiments were conducted on an NVIDIA GTX 480 GPU, programmed with CUDA 3.2 runtime API, hosted by an Intel Core 2 Duo CPU @ 2.66GHz. Table 7 shows the performance of the different stages of our method. The feature extraction part includes the thresholding and randomization of the resulting voxels. The visibility test portion is for testing the cluster normal directions at all views and also includes the splatting-based occlusion number computation. The most time-consuming part is the visibility test which would be much slower without GPU acceleration. The total processing time is about 2-8 times faster than a volume rendering of 1,800 views with a

fixed transfer function. Table 8 shows the optimal number of viewpoints computed by the ant colony-based SCP solver in 10 seconds and the number it finally converges to (marked by ∞). For practical datasets, the voxels were filtered at above 3% of the maximum scalar value to remove noise, and the average cluster width was in the range of 10-20 voxels.

Table 7. The performance of different stages of our viewpoint suggestion pipeline

Dataset	Datasize	Feature Extraction / K-Means Clustering / Visibility Test	Rendering Time / Speedup
Cube	256x256x256	0.9s / 0.9s / 7.3s	18s / 2.0×
Cube + Text	256×256×256	1.2s / 0.5s / 7.9s	18s / 1.9×
Tooth	256×256×161	1.2s / 1.3s / 8.9s	93s / 8.2×
Engine	256×256×110	1.1s / 0.6s / 10.4s	53s / 4.4×
Blunt Fin	256×128×64	0.7s / 0.6s / 8.2s	26s / 2.7×
Carp	256×256×512	2.5s / 2.5s / 9.2s	87s / 6.1×

5.7 Evaluations

We performed a simple user study to evaluate the effectiveness of our iView interface. For this, we invited 9 graduate students, all familiar with volume rendering and transfer function design. At all time, the subjects were permitted to use the 1D or 2D transfer function editor and choose any preferred view to look at the volume, with a fixed front-light and using the track-ball interface. The only testing condition was that they either had access to our entropy map or not. In the latter case the track-ball surface was simply left blank.

Table 8. The problem size of the K-Means clustering and the minimum number of views found by the SCP solver

Dataset	Voxels	Initial k / Resulting k	Averaged Cluster Width	Views (10s / ∞)
Cube	968	50 / 47	2.7	4 / 4
Cube + Text	1278	60 / 58	2.8	5 / 5
Tooth	170228	100 / 79	12.9	7 / 6
Engine	529050	120 / 111	16.4	7 / 6
Blunt Fin	65053	50 / 50	10.9	6 / 5
Carp	331894	80 / 69	16.9	5 / 5

Each subject/user would render two different dataset in turn – the tooth and the carp. For each dataset, a user was asked to construct two galleries (that is, select a set of viewpoints) that would best expose the salient information of the given dataset. The first gallery was always constructed with no entropy-map guidance, while for the second gallery this guidance was available. Prior to using the system, each user was trained on how to navigate with the entropy sphere (if provided), how to interpret its data, how to observe the clustering results and also how to use keys to add/delete views from the gallery.

We compared the views selected with and without the iView guidance. We found that in general users would pick fewer views without guidance. The mean of the difference between two sets of views was 0.90 (tooth) and 0.56 (carp). We used the dependent t-test for paired samples to analyze the view numbers with the hypothesis that that two means (with/without entropy map) are identical. The p-values for tooth and carp were 0.003 and 0.05, respectively. Thus, the fact that users would consistently pick fewer views without guidance indicates that iView helps users in locating commonly overlooked regions.

It was also interesting to observe that no gallery generated without guidance would include all of the top three views found with guidance (or with the SCP solver). There were often redundant views in the uninformed gallery or views with low entropy. To capture this behavior more quantitatively, we measured a given gallery's information coverage by the sum of entropy left in the map after gallery composition. When a user was allowed to use the map, the sum of the entropy left dropped from 51% to 24% for the tooth and from 37% to 19% for the carp, on average. Likewise, the percentage of entropy covered per view increased from 11% to 15% for the tooth and from 14% to 16% for the carp. This demonstrates that our navigation interface can clearly help users to optimize a set of viewing positions and with it the information seen.

Chapter 6. Conclusions

6.1 Summary

In this dissertation, we present our recent research results on an integrated reconstruction and visualization framework, including reconstruction optimizer, resampling verification and viewport advisor. This framework is applicable to many CT scanning scenario as occurring in medical, industrial, and security scanning applications. It contributes to the research of medical imaging and volume visualization. Theoretically, it helps the understanding of parameter optimization and error propagation in CT data processing. Practically, our GPU-accelerated framework has great potential in many valuable real-world applications.

Our contributions in the CT reconstruction and visualization domain include:

- We develop cache-aware algorithms to further accelerate CT reconstruction process. While in back-projection and in bi-lateral filter the improvement over straightforward GPU implementation are marginal, the 4-fold speedup achieved in NLM filter implementation makes it a practical candidate for low-dose CT reconstruction.
- We propose a CT reconstruction optimizer can automatically search parameters for iterative CT reconstruction. The training algorithm can efficiently learn reconstruction parameters to apply on similar data. It incorporates an interactive parameter visualization interface that can help researchers to understanding of X-ray dose, image quality and reconstruction speed.
- We present a systematic resampling verification method which is tightly coupled with the data generation process itself. It bounds the loss of accuracy associated with off-grid sample interpolation during rendering. Our method is able to certify a CT reconstructed volume for use with a given interpolation filter, explicitly specifying the maximum error that might occur in the

rendering. It uses a mixed-resolution volume encoding computed in a pre-process. This representation can maintain the advantages of real-time rendering, and at the same time gives the verifiable results.

- We devise a viewport advisor which can suggests users promising viewpoints for volume visualization prior to transfer function design activities. This transfer function neutral approach cuts down on the volume exploration effort since it selects potentially interesting views before laborious transfer function exploration begins. This is cognitively less challenging than changing viewpoint and transfer function at the same time. It can automatically suggest a set of optimal views automatically to compose an overview gallery.

6.2 Future Works

There are some research topics which directly extend our work. The CT reconstruction can be further accelerated in CPU/GPU clusters. We used ant colony optimization in CT reconstruction parameter searching. The parameter searching is an extremely computational intensive that could take a week to run in a single workstation with advanced GPUs. Recently cloud computing businesses provide infrastructure as a service (ISSA), which make high-end GPU clusters affordable. For example, currently Amazon Elastic Compute Cloud (Amazon EC2) rate for their CPU/GPU clusters is about a dollar per hour. These advancements bring a huge need for designing and developing software as a service (SAAS), which open a new direction for CT research and development. With the growing accessibility to GPU clusters, we are also planning to extend the current parameter search framework to include full 3D reconstruction and study parameter optimization in different scanning scenarios, for example, patient size, X-ray tube's voltage.

Our parameter search is based on an automatic quality observer in place of a human observer. We could improve the automatic quality observer by studying the link between the quality metric and the clinical need. Perceptual quality metrics are good candidates for this purpose. Ultimately, the goal is to develop an advanced quality metric which would pass doctors' validation.

Our current parameter visualization interface is designed to be available after parameter searching. For the future work, we could involve user interaction during the optimization stage. This more sophisticated interface will be an online monitoring tool. It can visualize the searching process while the computation is going on in the background. It can also open the doors for dynamic optimization functionality. The current web service interface is well adapted to SAAS model and we plan to work on additional support for gestures in mobile device.

There are also many future directions on the volume visualization side. Future work on resampling will adapt the verification concepts for more efficient grids, such as BCC. We would like to further evaluate and incorporate other errors occurring in the rendering such as shading, gradient estimation, transfer functions, perception, and the like.

For future work in viewport suggestion, we plan to extend our feature descriptor to other known metrics, such as suggestive contours (where the derivative of the normal vector is 0) and other well-known descriptors from computer vision: the multi-scale Harris detector [32] or SIFT [54], which we used already in other work for feature detection [62]. Further, we also believe that the silhouette metric would be a promising candidate for dynamic flow visualization and we plan to research this more thoroughly.

References

[1] S. Abbasi and F. Mokhtarian, "Automatic view selection in multi-view object recognition," in *Proceedings of International. Conference on Pattern Recognition*, vol. 1, pp. 13-16, 2000.

[2] A. Amirkhanov, C. Heinzl, M. Reiter, and E. Gröller, "Visual optimality and stability analysis of 3DCT scan positions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1477-1487, 2010.

[3] M. Artner, T. Möller, I. Viola, and M.E. Gröller, "High-quality volume rendering with resampling in the frequency domain," in *Proceedings of Eurographics/IEEE VGTC Symposium on Visualization*, pp. 85-92, 2005.

[4] J. E. Beasley, "OR-library: distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, no. 11, pp. 1069-1072, 1990.

[5] J. Beyer, M. Hadwiger, T. Möller, and L. Fritz, "Smooth mixed-resolution GPU volume rendering," in *Proceedings of IEEE International Symposium on Volume and Point-Based Graphics*, pp. 163-170, 2008.

[6] P. S. Blaer and P. K. Allen, "View planning and automated data acquisition for 3-D modeling of complex sites", *Journal of Field Robotics*, vol. 26, no. 11, pp. 865-891, 2009.

[7] U. Bordoloi and H.-W. Shen, "View selection for volume rendering," In *Proceedings of the IEEE Visualization*, pp. 487-494, 2005.

[8] R. Bracewell, *The Fourier Transform and its Applications*. 3rd edition, McGraw-Hill, 1999.

[9] A. Buades, B. Coll, and J. M. Morel, "A non-local algorithm for image denoising," *Computer Vision and Pattern Recognition*, pp. 60-65, 2005.

[10] M.-Y. Chan, H. Qu, K.-K. Chung, W.-H. Mak, and Y. Wu, "Relation-aware volume exploration pipeline," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1683-1690, 2008.

[11] M.-Y. Chan, Y. Wu, W.-H. Mak, W. Chen, and H. Qu, "Perception-based transparency optimization for direct volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1283-1290, 2009.

[12] L. Condat, T. Blu, and M. Unser, "Beyond interpolation: optimal reconstruction by quasi-interpolation," in *Proceedings of IEEE. International Conference on Image Processing*, pp. 33–36, 2005.

[13] C. Correa and K.-L. Ma, "Size-based transfer functions: a new volume exploration technique," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1380-1387, 2008.

[14] C. Correa and K.-L. Ma, "The occlusion spectrum for volume classification and visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1465-1472, 2009.

[15] C. Correa and K.-L. Ma, "Visibility histograms and visibility-driven transfer functions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 2, pp. 192-204, 2011.

[16] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann, "GigaVoxels: ray-guided streaming for efficient and detailed voxel rendering," in *Proceedings of ACM Symposium on Interactive 3D Graphics and Games*, pp. 15-22, 2009.

[17] B. Csébfalvi, "An evaluation of prefiltered reconstruction schemes for volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 289-301, 2008.

[18] B. Csébfalvi and B. Domonkos, "Frequency-domain upsampling on a body-centered cubic lattice for efficient and high-quality volume rendering," in *Proceedings of Vision, Modeling, and Visualization Workshop*, pp. 225-232, 2009.

[19] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 848-855, 2003.

[20] W. Degen, "Sharp error bounds for piecewise linear interpolation of planar curves," *Computing*, vol. 79, no. 2, pp. 143-151, 2007.

[21] M. do Carmo, *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[22] M. Dorigo, G. DiCaro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life*, vol. 5, no. 2, pp.137–172, 1999.

[23] K. Engel, M. Hadwiger, C. Rezk-Salama, and J. Kniss, *Real-Time Volume Graphics*. AK Peters Ltd, 2006.

[24] A. Entezari and T. Möller, "Extensions of the Zwart-Powell Box Spline for Volumetric Data Reconstruction on the Cartesian Lattice," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1337-1344, 2006.

[25] A. Entezari, T. Meng, S. Bergner, and T. Möller, "A granular three dimensional multiresolution transform," in *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization*, pp. 267-274. 2006.

[26] T. Etiene, C. Scheidegger, L. Nonato, R. Kirby, and C. Silva, "Verifiable visualization for isosurface extraction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1227-1234, 2009.

[27] W. Fang, K. -K. Lau, M. Lu, X. Xiao, C. Kit Lam, P. Y. Yang, B. He, Q. Luo, P. V. Sander, and K. Yang, "Parallel data mining on graphics processors," *Technical Report*, HKUST-CS08-07, 2008.

[28] T. Fogal and J. Krüger, "Tuvok - an architecture for large scale volume rendering," In *Proceedings of Vision, Modeling, and Visualization workshop*, pp. 139-146, 2010.

[29] L. A. Feldkamp, L. C. Davis, and J. W. Kress, "Practical cone-beam algorithm," *Journal of the Optical Society of America*, vol. 1, no. A6, 612-619, 1984.

[30] S. Fleishman, D. Cohen-Or, and D. Lischinski, "Automatic camera placement for image-based modeling," *Computer Graphics Forum*, vol. 19, no. 2, pp. 101-110, 2000.

[31] M. Frigo and S. Johnson, “The design and implementation of FFTW3,” in *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216-231, 2005.

[32] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Proceedings of 4th Alvey Vision Conference*, pp. 147-151, 1988.

[33] L. Hillebrand, R. Lapp, Y. Kyriakou, and W. Kalender, “Interactive GPU-accelerated image reconstruction in cone-beam CT,” in *Proceedings of SPIE*, vol. 7258, pp. 72582A-72582A-8, 2009.

[34] J. Huang, J. Ma, N. Liu, H. Zhang, Z. Bian, Y. Feng, Q. Feng, and W. Chen, “Sparse angular CT reconstruction using non-local means based iterative-correction POCS,” *Computers in Biology and Medicine*, vol. 41, no. 4, pp. 195-205, 2011.

[35] http://www.imagevis3d.org, ImageVis3D: A Real-time Volume Rendering Tool for Large Data. Scientific Computing and Imaging Institute (SCI).

[36] X. Jia, B. Dong, Y. F. Lou, and S. B. Jiang, “GPU-based iterative cone-beam CT reconstruction using tightframe regularization,” *Physics in Medicine and Biology*, vol. 56, no. 13, pp. 3787-3807, 2011.

[37] R. Kähler, M. Simon, and H. Hege, “Interactive volume rendering of large sparse data sets using adaptive mesh refinement hierarchies,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 3, pp. 341-351, 2003.

[38] R. Kähler, J. Wise, T. Abel, and H. Hege, “GPU-assisted raycasting for cosmological adaptive mesh refinement simulations,” in *Proceedings of Eurographics/IEEE Workshop on Volume Graphics*, pp. 103-110, 2006.

[39] R. Karp, “Reducibility among combinatorial problems,” *Complexity of Computer Computations*, pp. 85-103, 1972.

[40] A. Kaufman, S. Lakare, K. Kreeger, and I. Bitter, “Virtual colonoscopy,” *Communication of the ACM*, vol. 48, no. 2, pp. 37-41, 2005.

[41] B. Keck, H. Hofmann, H. Scherl, M. Kowarschik, and J. Hornegger, “GPU-accelerated SART reconstruction using the CUDA programming environment”, In *Proceedings of SPIE*, vol. 7258, pp. 7258-72582B, 2009.

[42] D. J. Ketchen and C. L. Shook, “The application of cluster analysis in strategic management research: an analysis and critique,” *Strategic Management Journal*, vol. 17, no. 6, pp. 441-458, 1996.

[43] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Möller, “Curvature-based transfer functions for direct volume rendering: methods and applications,” in *Proceedings of IEEE Visualization*, pp. 513-520, 2003.

[44] R. Kirby and C. Silva, “The need for verifiable visualization,” *IEEE Computer Graphics and Applications*, vol. 28, no. 5, pp. 78-83, 2008.

[45] J. Kniss, G. Kindlmann, and C. Hansen, “Multi-dimensional transfer functions for interactive volume rendering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 270-285, 2002.

[46] P. Kohlmann, S. Bruckner, A. Kanitsar, and M. E. Gröller, "LiveSync: deformed viewing spheres for knowledge-based navigation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1544-1551, 2007.

[47] J. Krüger, J. Schneider, and R. Westermann, "ClearView: an interactive context preserving hotspot visualization technique," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 941-948, 2006.

[48] E. LaMar, B. Hamann, and K. Joy, "Multiresolution techniques for interactive texture-based volume visualization," in *Proceedings of IEEE Visualization*, pp. 355-361, 1999.

[49] P. La Riviere, J. Bian, and P. Vargas, "Penalized-likelihood sinogram restoration for computed tomography," *IEEE Transactions on Medical Imaging*, vol. 25, no. 8, pp. 1022-36, 2006.

[50] A. Li, K. Mueller, and T. Ernst, "Methods for efficient, high quality volume resampling in the frequency domain," in *Proceedings of IEEE Visualization*, pp. 3-10, 2004.

[51] Z. Li, L. Yu, J. Trzasko, J. Fletcher, C. McCollough, and A. Manduca, "Adaptive nonlocal means filtering based on local noise level for CT denoising," in *Proceedings of SPIE*, vol. 8313, pp. 83131H, 2012.

[52] P. Ljung, C. Lundström, A. Ynnerman, and K. Museth, "Transfer function based adaptive decompression for volume rendering of large medical data sets," in *Proceedings of IEEE Volume Visualization and Graphics Symposium*, pp. 25-32, 2004.

[53] P. Ljung, C. Lundström, and A. Ynnerman, "Multiresolution interblock interpolation in direct volume tendering," in *Proceedings of Eurographics/IEEE VGTC Symp on Visualization*, pp. 259-266, 2006.

[54] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[55] R. Maciejewski, I. Woo, W. Chen, and D. Ebert, "Structuring feature space: a non-parametric method for volumetric transfer function generation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1473-1480, 2009.

[56] W.-H. Mai, Y. Wu, M.-Y. Chan, and H. Qu, "Visibility-aware direct volume rendering," *Journal of Computer Science and Technology*, vol. 26, no. 2, pp. 217-228, 2011.

[57] T. Malzbender, "Fourier volume rendering," *ACM Transactions on Graphics*, vol. 12, no. 3, pp.233-250, 1993.

[58] S. Marchesin and G.C. de Verdiere, "High-quality, semi-analytical volume rendering for AMR data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no.6, pp. 1611-1618, 2009.

[59] S. Marschner and R. Lobb, "An evaluation of reconstruction filters for volume rendering," in *Proceedings of IEEE Visualization*, pp. 100-107, 1994.

[60] C. Men, X. Gu, D. Choi, A. Majumdar, Z. Zheng, K. Mueller, and S. B. Jiang, "GPU-based ultra fast IMRT plan optimization," *Physics in Medicine and Biology*, vol. 54, pp. 6565-6573, 2009.

[61] T. Möller, R. Machiraju, K. Mueller, and R. Yagel, "Evaluation and design of filters using a Taylor series expansion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 184-199, 1997.

[62] J. Nam, M. Mauer, and K. Mueller, "High dimensional feature descriptors to characterize volumetric data," in *Proceedings of Knowledge-Assisted Visualization Workshop*, 2008.

[63] P. Noël, A. Walczak, J. Xu, J. Corso, K. Hoffmann, and S. Schafer, "GPU-based cone beam computed tomography," *Computer Methods and Programs in Biomedicine*, vol. 98, no. 3, pp. 271-277, 2010.

[64] Y. Okitsu, F. Ino, and K. Hagihara, "High-performance cone beam reconstruction using CUDA compatible GPUs", *Parallel Computing*, vol. 36, no. 2-3, pp. 129-141. 2010.

[65] E. Papenhausen, Z. Zheng, and K. Mueller, "GPU-accelerated back-projection revisited: squeezing performance by careful tuning," In *Proceedings of Workshop on High Performance Image Reconstruction*, pp. 19-22, 2011.

[66] D. Pelleg and A. Moore, "X-means: extending k-means with efficient estimation of the number of clusters," In *Proceedings of the 17th International Conference on Machine Learning*, pp. 727-734, 2000.

[67] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Transactions of Pattern Analysis and Maching Intelligence*, vol. 21, no. 10, pp. 1016-1030, 1999.

[68] M. Rahoual, R. Hadji, and V. Bachelet, "Parallel ant system for the set covering problem," *Lecture Notes in Computer Science*, vol. 2463, pp. 249-297, 2002.

[69] P. Rautek, B. Csébfalvi, S. Grimm, S. Bruckner, and M.E. Gröller, "D^2VR: high-quality volume rendering of projection-based volumetric data," in *Proceedings of Eurographics/IEEE VGTC Symp on Visualization*, pp. 211-218, 2006.

[70] Z. Ren, Z. Feng, L. Ke, and Z. Zhang, "New ideas for applying ant colony optimization to the set covering problem," *Computers and Industrial Engineering*, vol. 58, no. 4, pp.774-784, 2010.

[71] C. Rohkohl, B. Keck, H. G. Hofmann, and J. Hornegger, "RabbitCT - an open platform for benchmarking 3D cone-beam reconstruction algorithms," *Medical Physics*, vol. 36, no. 9, pp. 3940-3944, 2009.

[72] W. Scott, G. Roth, and J. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Computing Surveys*, vol. 35, no. 1, pp. 64-96, 2003.

[73] P. Sereda, A. Vilanova, and F. A. Gerritsen, "Automating transfer function design for volume rendering using hierarchical clustering of material boundaries," In *Proceedings of Eurographics/IEEE VGTC Symp on Visualization* , pp. 243-250, 2006.

[74] H. Scherl, B. Keck, M. Kowarschik, and J. Hornegger, "Fast GPU-based CT reconstruction using the common unified device architecture (CUDA)," *Nuclear Science Symposium Medical Imaging Conference*, vol. 6, pp. 4464-4466, 2007.

[75] E. Y. Sidky, C.-M. Kao, and X. Pan, "Accurate image reconstruction from few-views and limited-angle data in divergent beam CT," *Journal of X-Ray Science and Technology*, vol. 14, no. 2, pp. 119-139, 2006.

[76] E. Y. Sidky and X. Pan, "Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization," *Physics in Medicine and Biology*, vol. 53, no. 17, pp. 4777-4807, 2008.

[77] J. Smith and P. Gossett, "A flexible sampling-rate conversion method," in *Proceedings IEEE International Conference Acoustics, Speech, and Signal*, pp. 112-115, 1984. (Tutorial at http://ccrma.stanford.edu/~jos/resample).

[78] J. Song, Q. H. Liu, G. A. Johnson, and C. T. Badea, "Sparseness prior based iterative image reconstruction for retrospectively gated cardiac micro-CT," *Medical Physics*, vol. 34, no. 11, pp. 4476-4483, 2007.

[79] N. Sudarsanam, K. Singh, and C. Grimm, "Non-linear perspective widgets for creating multiple-view images," in *Proceedings of Symposium on Non-photorealistic Animation and Rendering*, pp. 69-79, 2008.

[80] P. Suetens, *Fundamentals of Medical Imaging*. Cambridge University Press, 2002.

[81] S. Takahashi, I. Fujishiro, Y. Takeshima, and T. Nishita, "A feature-driven approach to locating optimal viewpoints for volume visualization," in *Proceedings IEEE Visualization*, pp. 495-502, 2005.

[82] J. Tang, B. E. Nett, and G. H. Chen, "Performance comparison between total variation (TV)-based compressed sensing and statistical iterative reconstruction algorithms," *Physics in Medicine and Biology*, vol. 54, no. 19, pp. 5781-5804, 2009.

[83] P. Thevenaz, T. Blu, and M. Unser, "Interpolation revisited," *IEEE Transactions on Medical Imaging*, vol. 19, no. 7, pp. 739-758, 2000.

[84] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *IEEE International Conference on Computer Vision*, pp. 839-846, 1998.

[85] M. Unser, "Sampling – 50 Years after Shannon," in *Proceedings of the IEEE*, vol. 88, no. 4, pp. 569-587, 2000.

[86] P.-P. Vazquez, M. Feixas, M. Sbert, and W. Heidrich, "Viewpoint selection using view entropy," In *Proceedings of Vision Modeling and Visualization Conference*, pp. 273-280, 2001.

[87] P.-P. Vazquez, M. Feixas, M. Sbert, and W. Heidrich, "Automatic view selection using viewpoint entropy and its application to image-based modeling," *Computer Graphics Forum*, vol. 22, no. 4, pp. 689-700, 2003.

[88] I. Viola, A. Kanitsar, and M. E. Gröller, "Importance-driven feature enhancement in volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 4, pp. 408-418, 2005.

[89] L. Wang, Y. Zhao, K. Mueller, and A. E. Kaufman, "The magic volume lens: an interactive focus+context technique for volume rendering," in *Proceedings IEEE Visualization*, pp. 367-374, 2005.

[90] S. Wenhardt, B. Deutsch, J. Hornegger, H. Niemann, and J. Denzler, "An information theoretic approach for next best view planning in 3-D reconstruction," *International Conference on Pattern Recognition*, pp.103-106, 2006.

[91] R. Westermann and B. Sevenich, "Accelerated volume ray-casting using texture mapping," in *Proceedings of the conference on Visualization*, pp. 271-278, 2001.

[92] Y. Wu, K.-K. Chung, H. Qu, X Yuan, and S.C. Cheung, "Interactive visual optimization and analysis for RFID benchmarking," *IEEE Transactions Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1335-1342, 2009.

[93] F. Xu and K. Mueller, "GPU-accelerated D^2VR," in *Proceedings of Eurographics/ IEEE VGTC Workshop on Volume Graphics*, pp. 23-30, 2006.

[94] F. Xu and K. Mueller, "Real-time 3D computed tomographic reconstruction using commodity graphics hardware," *Physics in Medicine and Biology*, vol. 52, no. 12, pp. 3405-3419, 2007.

[95] F. Xu, W. Xu, M. Jones, B. Keszthelyi, J. Sedat, D. Agard, and K. Mueller, "On the efficiency of iterative ordered subset reconstruction algorithms for acceleration on GPUs," *Computer Methods and Programs in Biomedicine*, vol. 98, no. 3, pp. 261-270, 2010.

[96] W. Xu and K. Mueller, "Accelerating Regularized Iterative CT Reconstruction on Commodity Graphics Hardware (GPU)," *IEEE International Symposium on Biomedical Imaging (ISBI)*, 2009.

[97] W. Xu and K. Mueller, "A performance-driven study of regularization methods for GPU-accelerated iterative CT," in *Proceedings of Workshop on High Performance Image Reconstruction*, 2009.

[98] W. Xu and K. Mueller, "Evaluating popular non-linear image processing filters for their use in regularized iterative CT," *Nuclear Science Symposium Medical Imaging Conference*, 2010.

[99] W. Xu and K. Mueller, "Parameter space visualizer: an interactive parameter selection interface for iterative CT reconstruction algorithms" in *Proceedings of SPIE*, vol. 7625, pp. 76251Q, 2010.

[100] W. Xu and K. Mueller, "A reference image database approach for NLM filter-regularized CT reconstruction," in *Proceedings of Fully 3D Image Reconstruction in Radiology and Nuclear Medicine*, pp. 116-119, 2011.

[101] W. Xu and K. Mueller, "Using GPUs to learn effective parameter settings for GPU-accelerated iterative CT reconstruction algorithms," *GPU Computing Gems Emerald Edition*, Chapter 43, 2011.

[102] H. Yan, L. Cervino, X. Jia, and S. B. Jiang, "A comprehensive study on the relationship between image quality and imaging dose in low-dose cone beam CT," *Physics in Medicine and Biology*, vol. 57, no. 7, pp. 2063-2080, 2012.

[103] H. Yu and G. Wang, "A soft-threshold filtering approach for reconstruction from a limited number of projections," *Physics in Medicine and Biology*, vol. 55, no. 13, pp. 3905-3916, 2010.

[104] W. Zbijewski and F. Beekman, "Efficient Monte Carlo based scatter artifact reduction in cone-beam micro-CT," *IEEE Transactions on Medical Imaging*, vol. 25, vol. 7, pp. 817-827, 2006.

[105] R. Zhang, P.-S, Tsai, J.E. Cryer, and M. Shah, "Shape from shading: a survey," *IEEE Transactions Patten Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 690-706, 1999.

[106] Z. Zheng and K. Mueller, "VDVR: verifiable volume visualization for projection-based data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1515-1524, 2010.

[107] Z. Zheng, N. Ahmed, and K. Mueller, "iView: a feature clustering framework for suggesting informative views in volume visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1959-1968, 2011.

[108] Z. Zheng and K. Mueller, "Identifying sets of favorable projections for few-view low-dose cone-beam CT Scanning," in *Proceedings of The 11th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, pp. 314-317, 2011.

[109] Z. Zheng, W. Xu, and K. Mueller, "Performance tuning for CUDA-accelerated neighborhood denoising filters," In *Proceedings of workshop on High Performance Image Reconstruction*, pp. 52-55, 2011.