Fast Katsevich Algorithm Based on GPU for Helical Cone-Beam Computed Tomography

Guorui Yan, Jie Tian, *Fellow, IEEE*, Shouping Zhu, Chenghu Qin, *Member, IEEE*, Yakang Dai, Fei Yang, Di Dong, and Ping Wu

***Abstract*—Katsevich reconstruction algorithm represents a breakthrough for helical cone-beam computed tomography (CT) reconstruction, because it is the first exact cone-beam reconstruction algorithm of filtered backprojection (FBP) type with 1-D shift-invariant filtering. Although FBP-type reconstruction algorithm is effective, 3-D CT reconstruction is time-consuming, and the accelerations of Katsevich algorithm on CPU or cluster have been widely studied. In this paper, Katsevich algorithm is accelerated by using graphics processing unit, including flat-detector and curved-detector geometry in the case of helical orbit. An overscan formula is derived, which helps to avoid unnecessary overscan in practical CT scanning. Based on the overscan formula, a volume-blocking method in device memory is proposed. One advantage of the blocking method is that it can reconstruct large volume with high speed.**

***Index Terms*—Computed tomography (CT), graphics processing unit (GPU), Katsevich algorithm, overscan.**

I. Introduction

FELDKAMP *et al.* [1] first proposed a 3-D computed tomography (CT) algorithm of filtered backprojection (FBP) type, which is popular even now because of its efficiency. But it is an approximate algorithm, and it introduces image artifacts that will be aggravated with the increase of cone-beam angle. In 1993, a generalized Feldkamp algorithm was proposed by Wang *et al.* [2] in the case of helical cone-beam CT, and it is also an approximate algorithm. In recent years, several exact algorithms have been developed, including Katsevich algorithm [3], [4], Zou and Pan's PI-line algorithm [5], [6] for helical trajectory, and some exact algorithms for general trajectory [7]–[9]. Among them, Katsevich reconstruction algorithm, developed in 2002, is the first exact cone-beam reconstruction algorithm of FBP type that represents a breakthrough in exact CT reconstruction. Although Katsevich algorithm is FBP type, 3-D CT image reconstruction is computationally demanding at computational complexity of $O(NM^3)$, where N and M, respectively, stand for the number of projection views and volumetric pixels in one dimension. It has been reported how to accelerate Katsevich algorithm on CPU or cluster by Deng *et al.* [10], Yang *et al.* [11], and Fontaine and Lee [12]. As far as we know, the acceleration of Katsevich algorithm on graphics processing unit (GPU) has not been reported in the literature so far.

In the past several years, Feldkamp-Davis-Kress (FDK) acceleration algorithms on GPU have been widely developed. Cabral *et al.* [13] first implemented the accelerated CT reconstruction on nonprogrammable Silicon Graphics Inc. (SGI) workstation; Xu and Mueller [14] developed accelerated graphics GPU (AG-GPU) mechanism on the commercial GeForce 8800GTX GPU; Yang *et al.* [15] performed the backprojection using compute unified device architecture (CUDA) architecture; Yan *et al.* [16] implemented FDK reconstruction algorithm using 3-D texture and a cyclic render-to-texture (CRTT) technology. In this paper, we study the acceleration of Katsevich algorithm on GPU hardware using OpenGL and Cg framework. The filtering step is performed on CPU, and the most time-consuming step, backprojection, is performed on GPU, which is extended from circle orbit to helix and from flat detector to curved detector.

The other key contribution is that we derive an overscan formula for helical orbit, which is helpful to avoid unnecessary overscan and unnecessary dose in practical CT scanning. Based on the overscan formula, a volume-blocking method in GPU memory is proposed that can reconstruct large volume with high speed. During reconstruction using GPU hardware, either 3-D or a stack of 2-D textures can be used for storing the reconstruction volume. For 3-D texture, if the volume data is larger than the size of GPU memory, 3-D texture will not work. For 2-D texture, it is able to reconstruct volume data larger than the size of device memory. However, it will cost much more time, because data transfer between main memory and device memory, which is very time-consuming, will be quite frequent when device memory is less than the volume size. The blocking method based on overscan formula can solve the conflict between large volume and high speed, effectively, only if the device memory can hold volume for two overscan distance.

Manuscript received March 26, 2009; revised July 15, 2009 and September 16, 2009; accepted October 30, 2009. Date of publication December 11, 2009; date of current version July 9, 2010. This paper was supported by the Project for the National Basic Research Program of China (973) under Grant 2006CB705700, by the Changjiang Scholars and Innovative Research Team in University (PCSIRT) under Grant IRT0645, by the Chinese Academy of Sciences (CAS) Hundred Talents Program, by the CAS Scientific Research Equipment Developement Program under Grant YZ200766, by the Knowledge Innovation Project of the CAS under Grant KGCX2-YW-129 and Grant KSCX2-YW-R-262, by the National Natural Science Foundation of China under Grant 60532050, Grant 30672690, Grant 30600151, Grant 60621001, Grant 30873462, Grant 60910006, Grant 30970769, and Grant 30970771, by the Beijing Natural Science Fund under Grant 4071003, and by the Technology Key Project of Beijing Municipal Education Commission under Grant KZ200910005005.

G. Yan, S. Zhu, C. Qin, Y. Dai, F. Yang, D. Dong, and P. Wu are with the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: yangr@fingerpass.net.cn; zhusp@fingerpass.net.cn; qinch@fingerpass.net.cn; daiyk@fingerpass.net.cn; yangfei@fingerpass.net.cn; dongdi@fingerpass.net.cn; wuping@fingerpass.net.cn).

J. Tian is with the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the Life Science Center, Xidian University, Xi'an 710071, China (e-mail: tian@ieee.org, website: http://www.mitk.net).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITB.2009.2036368

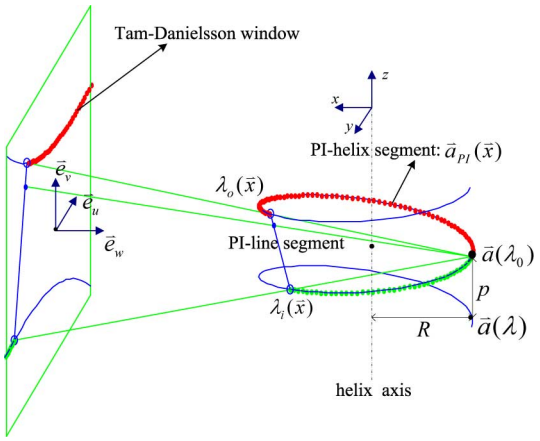


Fig. 1. Helix geometry. PI-line segment and Tam–Danielsson window.

The rest of this paper is organized as follows. In Section II, Katsevich algorithm is outlined briefly. In Section III, the implementation of Katsevich algorithm on GPU hardware using native geometries [17], which includes flat-detector geometry and curved-detector geometry, is introduced. In Section IV, experiments and results are analyzed. Finally, the conclusions of this paper are made.

II. Katsevich Algorithm

As shown in Fig. 1, a helical trajectory of rotation axis z, radius R, and pitch p is defined by

$$\vec{a}(\lambda) = [R\cos\,\lambda, R\sin\,\lambda, p\lambda/2\pi]. \tag{1}$$

U is a cylinder inside the spiral

$$U = \{\vec{x} \in R^3 : x^2 + y^2 < r^2\}, \qquad 0 < r < R \tag{2}$$

where $\vec{x} = (x, y, z)$. $f(\vec{x})$ represents the spatial function to be reconstructed, and it is zero outside the cylinder U. Cone-beam projection of $f(\vec{x})$ is defined as

$$D(\vec{a}(\lambda), \vec{\beta}) = \int_0^\infty f(\vec{a}(\lambda) + t\vec{\beta})\, dt \tag{3}$$

where $\vec{\beta}(\lambda, \vec{x}) = (\vec{x} - \vec{a}(\lambda))/(|\vec{x} - \vec{a}(\lambda)|$ or $\vec{\beta}$ for short.

Danielsson *et al.* [18] and Defrise *et al.* [19] have proved that for any $\vec{x} \in U$, there is one and only one PI-line passing through $\vec{x}$, and PI-line is defined as any line connecting two source points within one helical turn. Suppose $\lambda_i(\vec{x})$ and $\lambda_o(\vec{x})$ denote two endpoints, $I_{\rm PI}(\vec{x}) = [\lambda_i(\vec{x}), \lambda_o(\vec{x})]$ is the PI parametric interval [4], Katsevich has developed a reconstruction framework, described as follows:

$$\begin{aligned}(B_k f)(\vec{x}) = -\frac{1}{2\pi^2} \int_{\lambda_i(\vec{x})}^{\lambda_o(\vec{x})} \frac{1}{|\vec{x} - \vec{a}(\lambda)|} \\ \times \int_0^{2\pi} \frac{\partial}{\partial q} D(\vec{a}(q), \cos\,\gamma\vec{\beta}(\lambda, \vec{x}) \\ + \sin\,\gamma\vec{e}_k(\lambda, \vec{x}))\Big|_{q=\lambda} \frac{d\gamma}{\sin\,\gamma} d\lambda\end{aligned} \tag{4}$$

$$(B_k f)(\vec{x}) = \frac{1}{(2\pi)^3} \int_{R^3} B_k(\vec{x}, \vec{\xi}) \tilde{f}(\vec{\xi}) e^{-j\vec{\xi}\cdot\vec{x}} d\vec{\xi} \tag{5}$$

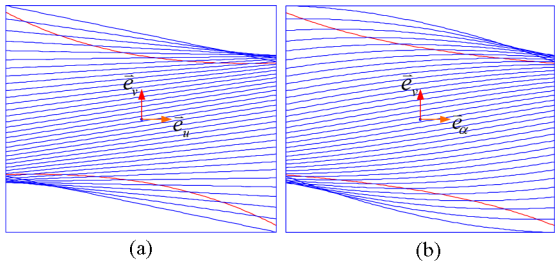


Fig. 2. Filtering is performed along the κ-lines. (a) κ-lines and Tam–Danielsson window on the flat detector. (b) κ-lines and Tam–Danielsson window on the curved detector.

where $\tilde{f}(\vec{\xi})$ is the Fourier transform of $f(\vec{x})$. $B_k(\vec{x}, \vec{\xi}) = \sum_{\lambda_j \in I_{PI}(\vec{x})} \mathrm{sgn}(\vec{\xi}\cdot\vec{a}'(\lambda_j))\mathrm{sgn}(\vec{\xi}\cdot\vec{e}_k(\lambda_j, \vec{x}))$ and $\vec{e}_k(\lambda, \vec{x})$ is an unit vector perpendicular to $\vec{\beta}(\lambda, \vec{x})$. Katsevich algorithm depends on how to choose $\vec{e}_k(\lambda, \vec{x})$. In other words, i.e., how to choose the filtering plane that passes through vector $\vec{e}_k(\lambda, \vec{x})$, $\vec{\beta}(\lambda, \vec{x})$, and the point $\vec{a}(\lambda)$, so as to make $B_k(\vec{x}, \vec{\xi})$ a constant for some k or some combination of k at almost every $\vec{\xi}$, where λ_j is the intersection of the filtering plane and the helical orbit.

Katsevich selected κ-planes as the filtering planes [4], [17]. A κ-plane is any plane that has three intersections with helical orbit at $\vec{a}(\lambda)$, $\vec{a}(\lambda + \psi)$, and $\vec{a}(\lambda + 2\psi)$ with $\psi \in [-\pi, \pi]$, and the plane is denoted by $\kappa(\lambda, \psi)$. Given a point $\vec{x}$, for any $\lambda \in I_{\rm PI}(\vec{x})$, there is one and only one the smallest $|\psi|$, which makes $\kappa(\lambda, \psi)$ passing through point $\vec{x}$ and $\lambda_i(\vec{x}) \le \lambda + 2\psi \le \lambda_o(\vec{x})$. In this circumstance, the normal vector of κ-plane is defined as $\vec{m}(\lambda, \beta)$.

Selecting κ-planes as filtering planes, i.e., choosing $\vec{e}_0(\lambda, \vec{x}) = \beta \times \vec{m}(\lambda, \beta)$, makes $B_k(\vec{x}, \vec{\xi}) = 1$ for almost all $\vec{\xi}$. Therefore, a reconstruction formula can be obtained as follows:

$$\begin{aligned}f(\vec{x}) = B_0 f = -\frac{1}{2\pi^2} \int_{\lambda_i(\vec{x})}^{\lambda_o(\vec{x})} \frac{1}{|\vec{x} - \vec{a}(\lambda)|} \\ \times \int_0^{2\pi} \frac{\partial}{\partial q} D(\vec{a}(q), \cos\,\gamma\vec{\beta}(\lambda, \vec{x}) \\ + \sin\,\gamma\vec{e}_0(\lambda, \vec{x}))\Big|_{q=\lambda} \frac{d\gamma}{\sin\,\gamma} d\lambda.\end{aligned} \tag{6}$$

The intersection of κ-plane and detector is called κ-line, along which the filtering is performed, as illustrated in Fig. 2. The detector region needed for reconstruction is slightly larger than the Tam–Danielsson window [18], [20], which is the projections of the upper and the lower helical trajectory onto the detector relative to X-ray source, as shown in Figs. 1 and 2.

III. Katsevich Algorithm Implementation

Noo *et al.* [17] introduced how to implement the Katsevich algorithm using native geometries that includes flat-detector geometry and curved-detector geometry. Our implementation is based on it. In this paper, filtering is performed on CPU using fastest Fourier transform in the west (FFTW) [21], and the backprojection is performed on GPU.

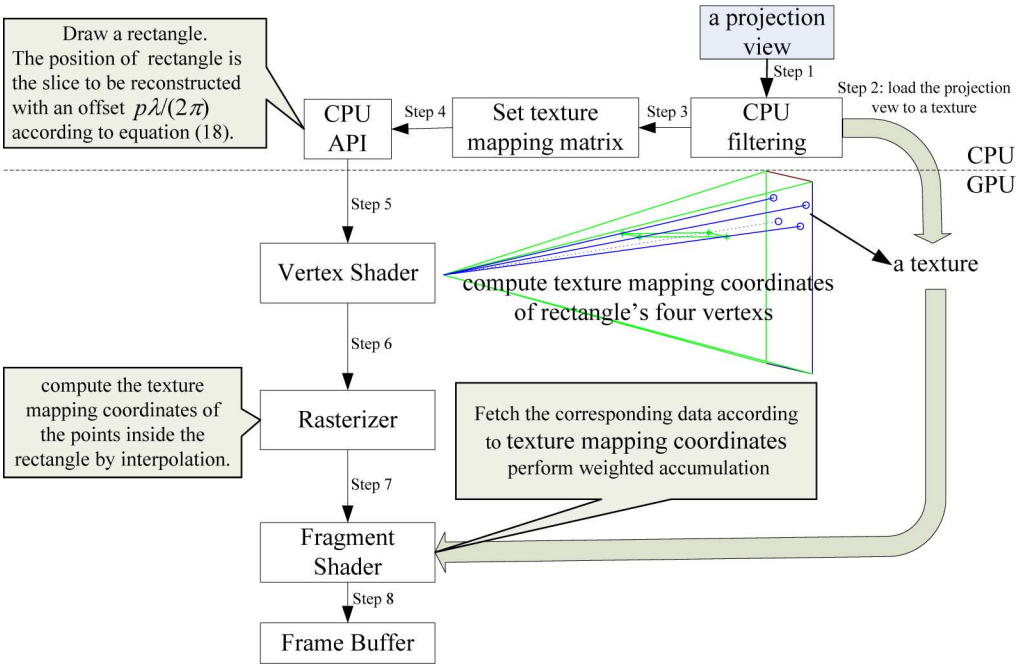


Fig. 3. Flowchart of the Katsevich algorithm's implementation. Filtering is performed on CPU, and backprojection is performed on GPU.

As illustrated in Fig. 3, filtering on CPU is first performed, and the filtered projection data are transferred to a texture, then texture-mapping matrix is set according to backprojection geometry. GPU pipeline is driven by drawing a rectangle, which is a slice to be reconstructed. In vertex shader, texture-mapping coordinates of the rectangle's four vertexes are computed, according to texture-mapping matrix set in step 2. Rasterization is completed automatically by GPU hardware. In this stage, the texture-mapping coordinates of the points inside rectangle are computed by interpolation. In fragment shader, the corresponding texture data are fetched, according to texture-mapping coordinates and weighted accumulation is performed. Finally, the results are written to the frame buffer. In this step, the technology render-to-texture or CRTT [16] can be used.

A. *Filtering on CPU*

In this section, we take flat-detector panel filtering as an example, and the curved-detector panel filtering is similar [17]. As illustrated in Fig. 4(a), the cone-beam measured data using flat detector are as follows:

$$D(\vec{a}(\lambda), u, v) = D(\vec{a}(\lambda), \vec{\beta}_f) \tag{7}$$

with

$$\vec{\beta}_f = \frac{1}{\sqrt{u^2+v^2+L^2}}(u\vec{e}_u(\lambda) + v\vec{e}_v(\lambda) - L\vec{e}_w(\lambda)). \tag{8}$$

The filtering is carried out as following steps [17].

F1: Derivative and weighting

$$D_1(\vec{a}(\lambda), u, v) = \frac{dD(\vec{a}(\lambda), \vec{\beta}_f)}{d\lambda}\text{weight}(u, v) \tag{9}$$

where $\text{weight}(u, v) = L/(\sqrt{L^2+u^2+v^2})$.

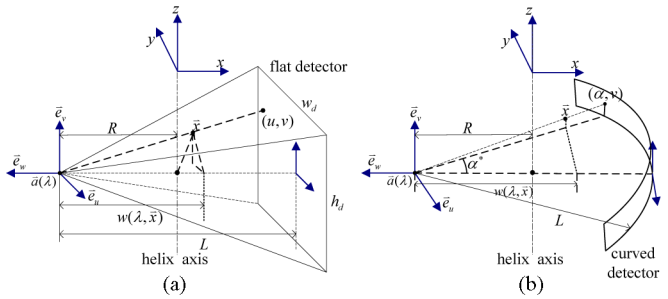


Fig. 4. Data acquisition geometry. The radius of helical trajectory is R. (a) Flat-detector geometry. The source-detector distance is L. (b) Curved-detector geometry. The radius of curved-detector plane is L.

F2: Compute the value on the κ-lines, according to (10), and the κ-lines on the flat-detector panel, as shown in Fig. 2(a).

$$D_2(\lambda, u, \psi) = D_1(\lambda, u, v(u, \psi)) \tag{10}$$

where $\psi \in [-\pi/2 - \alpha_m, \pi/2 + \alpha_m]$, $\alpha_m = \arcsin(r/R)$, and

$$v(u, \psi) = \frac{Lp}{2\pi R}\left(\psi + \frac{\psi}{\tan\psi}\frac{u}{L}\right). \tag{11}$$

F3: Hilbert transform

$$D_3(\lambda, u, \psi) = D_2(\lambda, u, \psi)h(u) \tag{12}$$

where

$$h(u) = -\int_{-\infty}^{\infty} d\sigma i\text{sgn}(\sigma)e^{i2\pi\sigma u} = \frac{1}{\pi u}. \tag{13}$$

F4: Backward height rebinning

$$D^F(\lambda, u, v) = D_3(\lambda, u, \widehat{\psi}(u, v)) \tag{14}$$

where $\widehat{\psi}(u, v)$ is the solution of (11) with the smallest absolute value.

Each step can be implemented within one loop. Since the Tam–Danielsson, κ-line forward index, κ-line backward index, and the weight to projection are independent of projection views in helical trajectory, which can be computed for one time and stored into tables.

B. Backprojection on GPU

The most time-consuming part of CT reconstruction is backprojection. After filtered on CPU, $D^F(\lambda,u,v)$ is backprojected to reconstruct $f(\vec{x})$, according to

$$f(\vec{x})=\frac{1}{2\pi}\int_{\lambda\in I_{\mathrm{PI}}(\vec{x})}d\lambda\frac{1}{w(\lambda,\vec{x})}D^F(\lambda,u(\lambda,\vec{x}),v(\lambda,\vec{x})). \tag{15}$$

Since the projection of $\vec{x}$ on detector will be outside the Tam–Danielsson window if $\lambda\notin I_{\mathrm{PI}}(\vec{x})$, setting the values outside the window to zeros is equivalent to integral within the interval $I_{\mathrm{PI}}(\vec{x})$. Therefore, the integral interval $I_{\mathrm{PI}}(\vec{x})$ can be simplified using Tam–Danielsson window as follows [17]:

$$f(\vec{x})=\frac{1}{2\pi}\int_{-\infty}^{\infty}d\lambda\frac{\chi(u,v)D^F(\lambda,u,v)}{w(\lambda,\vec{x})}\mid_{u=u(\lambda,\vec{x}),v=v(\lambda,\vec{x})} \tag{16}$$

with

$$\chi(u,v)=\begin{cases}0,\text{if } v>v_{\mathrm{top}}(u)+ad_v\\ \dfrac{v_{\mathrm{top}}(u)+ad_v-v}{2ad_v},\\ \text{if } v_{\mathrm{top}}(u)-ad_v<v<v_{\mathrm{top}}(u)+ad_v\\ 1,\text{if } v_{\mathrm{bottom}}(u)+ad_v<v<v_{\mathrm{top}}(u)-ad_v\\ \dfrac{v-v_{\mathrm{bottom}}(u)+ad_v}{2ad_v},\\ \text{if } v_{\mathrm{bottom}}(u)-ad_v<v<v_{\mathrm{bottom}}(u)+ad_v\\ 0,\text{if } v<v_{\mathrm{bottom}}(u)-ad_v\end{cases} \tag{17}$$

where v_{top} and v_{bottom} are the top and bottom of the Tam–Danielsson window, respectively, a is an adjustable parameter, and d_v is the thickness of the detector rows [17]. The production of $\chi(u,v)$ and $D^F(\lambda,u,v)$ can be incorporated into F4 on CPU.

The mapping between planar detector and a slice to be reconstructed is similar to projective texture mapping [22] in computer graphics. Cabral *et al.* [13] first implemented the accelerated CT reconstruction using projective texture mapping, and the most of following GPU accelerated algorithms were based on texture mapping [14], [16]. Based on these existing researches, the backprojection of the helical geometry will be described in the computer graphics form completely.

As illustrated in Fig. 4(a), $u(\lambda,\vec{x})$ and $v(\lambda,\vec{x})$ are the mapping coordinates on the flat detector of $\vec{x}$ relative to source $\vec{a}(\lambda)$, and $w(\lambda,\vec{x})$ is the distance between the source $\vec{a}(\lambda)$ and the projection of the point $\vec{x}$ to the $\vec{e}_w$ axis. We can obtain $u(\lambda,\vec{x}),v(\lambda,\vec{x})$, and $w(\lambda,\vec{x})$ in homogeneous,

$$\begin{aligned}\begin{bmatrix}u_h(\lambda,\vec{x})\\ v_h(\lambda,\vec{x})\\ 0\\ w_h(\lambda,\vec{x})\end{bmatrix}&=P\times V\times M\times\begin{bmatrix}x\\ y\\ z-\frac{p}{2\pi}\lambda\\ 1\end{bmatrix}\\ &=\begin{bmatrix}L&0&0&0\\ 0&L&0&0\\ 0&0&0&0\\ 0&0&-1&0\end{bmatrix}\begin{bmatrix}0&-1&0&0\\ 0&0&1&0\\ -1&0&0&-R\\ 0&0&0&1\end{bmatrix}\\ &\quad\times\begin{bmatrix}\cos\lambda&\sin\lambda&0&0\\ -\sin\lambda&\cos\lambda&0&0\\ 0&0&1&0\\ 0&0&0&1\end{bmatrix}\begin{bmatrix}x\\ y\\ z-\frac{p}{2\pi}\lambda\\ 1\end{bmatrix}\end{aligned} \tag{18}$$

i.e., $u(\lambda,\vec{x})=(u_h(\lambda,\vec{x}))/(w_h(\lambda,\vec{x})), v(\lambda,\vec{x})=(v_h(\lambda,\vec{x}))/(w_h(\lambda,\vec{x}))$, and $w(\lambda,\vec{x})=w_h(\lambda,\vec{x})$.

The model matrix M corresponds to the rotation of a cone-beam CT. The view matrix V reveals the distance between X-ray source and rotation center, and transforms the volume space to the source-detector space. The matrix P is a perspective transform. Since texture index is usually in $[0,1]$, a scaling and translation matrix TS is needed, which is as follows:

$$TS=\begin{bmatrix}\frac{1}{w_d}&0&0&0.5\\ 0&\frac{1}{h_d}&0&0.5\\ 0&0&1&0\\ 0&0&0&1\end{bmatrix}. \tag{19}$$

Therefore, the texture-mapping matrix of flat-detector panel is $TS\times P\times V\times M$, and the position of the rectangle, drawn in Fig. 3, is the slice to be reconstructed with an offset $(p/2\pi)\lambda$.

Compared to flat-detector geometry, the texture-mapping matrix of curved-detector geometry is not so direct, as shown in Fig. 4(b), $\alpha(\lambda,\vec{x})$ and $v(\lambda,\vec{x})$ are the mapping coordinates of $\vec{x}$ on the curved detector relative to source $\vec{a}(\lambda)$. Change the perspective matrix P to

$$P^c=\begin{bmatrix}1&0&0&0\\ 0&L&0&0\\ 0&0&0&0\\ 0&0&-1&0\end{bmatrix} \tag{20}$$

and obtain

$$\begin{bmatrix}u_h^c(\lambda,\vec{x})\\ v_h^c(\lambda,\vec{x})\\ 0\\ w_h^c(\lambda,\vec{x})\end{bmatrix}=P^c\times V\times M\begin{bmatrix}x\\ y\\ z-\frac{P}{2\pi}\lambda\\ 1\end{bmatrix}. \tag{21}$$

Set the texture-mapping matrix of curved-detector panel to $P^c\times V\times M$ and compute the mapping of point $\vec{x}$ in fragment shader by

$$\alpha(\lambda,\vec{x})=\arctan\left(\frac{u_h^c(\lambda,\vec{x})}{w_h^c(\lambda,\vec{x})}\right) \tag{22}$$

$$v(\lambda,\vec{x})=\frac{v_h^c(\lambda,\vec{x})\cos(\alpha(\lambda,\vec{x}))}{w_h^c(\lambda,\vec{x})}. \tag{23}$$

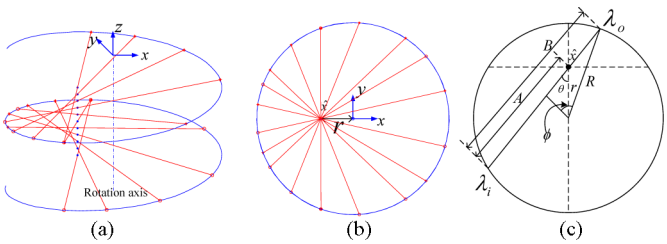


Fig. 5. (a) Line is parallel to z-axis and deviates z-axis at a distance of r. (b) and (c) Projection to $x \times y$ plane.

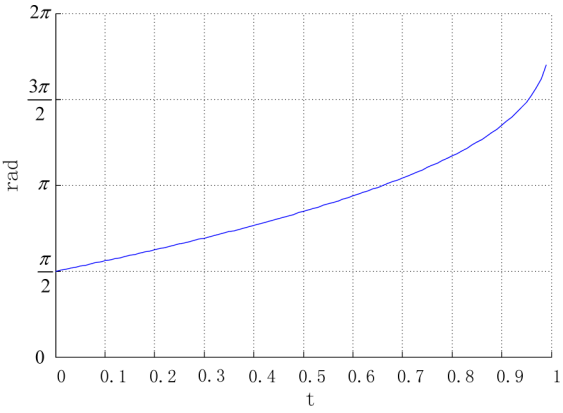


Fig. 6. Overscan for varying t.

TABLE I
OVERSCAN FOR $t = 1/3, t = 1/2$, AND $t = 4/5$

t	1/3	1/2	4/5
overscan (rad)	2.255	2.666	3.683
overscan (pitch)	$0.3588p$	$0.4243p$	$0.5861p$

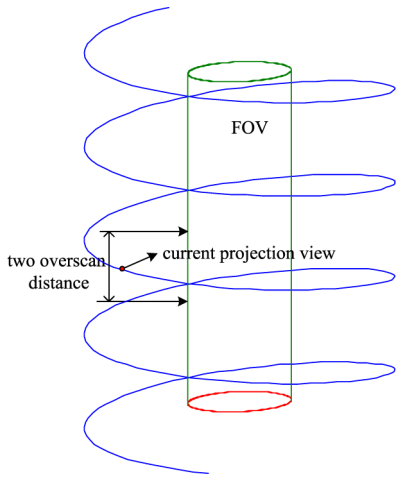


Fig. 7. Valid field for a projection view.

C. Overscan Formula and Volume Blocking in Device Memory

1) Overscan Formula: Some overscan for reconstructing a field of view (FOV) is necessary. As shown in (6), the useful scanning range for a point $\vec{x} = (x, y, z)$ is $I_{\mathrm{PI}}(\vec{x}) = [\lambda_i(\vec{x}), \lambda_o(\vec{x})]$, which consists of the overscan $\lambda_{\vec{x}} - \lambda_i(\vec{x})$ below the point $\vec{x}$ and $\lambda_o(\vec{x}) - \lambda_{\vec{x}}$ above the point $\vec{x}$, where $\lambda_{\vec{x}} = 2\pi z/p$. What is the overscan for reconstructing a volume? Consider a line that is parallel to z-axis and deviate z-axis at a distance of r, as illustrated in Fig. 5(a). The line and the PI-lines passing through the line are projected to $x \times y$ plane, and the projection of the line is denoted by $\hat{x}$, as shown in Fig. 5(b).

As illustrated in Fig. 5(c), the below and above overscan are equal due to symmetry, and the projections of the PI-lines passing through the line fulfill the circle;therefore, the overscan can be derived, as shown in (24) at the bottom of this page.

It is shown that the overscan is a function of $t = r/R$, and is increasing for $t \in (0, 1)$. Fig. 6 illustrates that the overscan varies with t, and some special values for $t = 1/3, t = 1/2$, and $t = 4/5$ are listed in Table I. It is shown that larger ratio t requires larger overscan.

2) Volume Blocking in Device Memory: As illustrated earlier, not all projections are useful for a slice to be reconstructed. For a projection view, it is only valid for the slices within one overscan distance to the projection, as illustrated in Fig. 7, and we call these slices as the projection view's valid field. Therefore, the memory allocated on the GPU device just need to store the volume for two overscan distance, instead of the whole FOV volume. The slice number allocated on device memory is

$$\mathrm{Slice}_{\mathrm{GPU}} = \frac{2D_{\mathrm{overscan}}}{dz} \tag{25}$$

where dz is the thickness of the slice and D_{overscan} is the overscan distance in pitch form.

As illustrated in Fig. 8, first, we allocate two-overscan-distance device memory. With the increase of projection position, there will be a slice no longer in the projection's valid

$$\begin{aligned}
\text{overscan} &= \max_{0\le\theta\le\pi/2} (2\pi - 2\phi)\frac{A}{B} \\
&= \max_{0\le\theta\le\pi/2} \frac{(2\pi - 2\arccos((r/R)\sin\theta))(r\cos\theta + R\sin(\arccos((r/R)\sin\theta)))}{2R\sin(\arccos(r/R\sin\theta))} \\
&= \max_{0\le\theta\le\pi/2} \frac{(\pi - \arccos((r/R)\sin\theta))(r\cos\theta + \sqrt{R^2 - r^2\sin^2\theta})}{\sqrt{R^2 - r^2\sin^2\theta}} \\
&= \max_{0\le\theta\le\pi/2} \frac{(\pi - \arccos(t\sin\theta))(t\cos\theta + \sqrt{1 - t^2\sin^2\theta})}{\sqrt{1 - t^2\sin^2\theta}} \left(0 < t = \frac{r}{R} < 1\right).
\end{aligned} \tag{24}$$

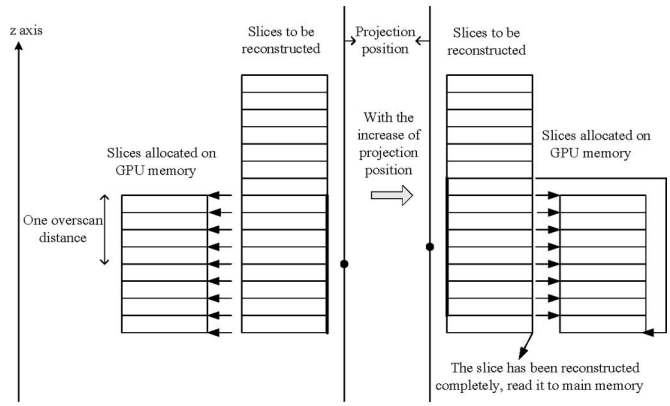


Fig. 8. Volume-blocking method in device memory.

TABLE II
SIMULATION PARAMETERS

Detector pixels(column by row)	672×128
Pixel size	0.01×0.01
Radius of the helix *R*	3
The pitch of the helix *p*	0.7
Source-detector distance *L*	6
The maximum FOV radius *r*	1
Number of projections (step: 1°)	1288

field, which means that this slice is completely reconstructed. Then, the slice data are transferred from device memory to main memory, and corresponding device memory can be used for reconstructing a new slice, as illustrated in Fig. 8.

The memory allocated on the GPU device just need to store the volume for two overscan distance, instead of the whole FOV volume, in which way, the limited device memory is saved, and fast reconstruction of large volume is possible.

IV. EXPERIMENTS AND RESULTS

Our experiments are performed on a 2.66-GHz dual-core Intel PC with 2 GB RAM hosting a NVIDIA GeForce 8800GTX card. The projections are analytically calculated from 3-D Shepp–Logan phantom, and the reconstruction FOVs are all $[-1, 1]$ in x, y, and z axes. In this paper, the algorithm is implemented for an ideally aligned detector. The geometrical simulation parameters are listed in Table II, where the pixel size for flat-detector geometry and curved-detector geometry are both 0.01×0.01. Different volume datasets are reconstructed from the projections, and single floating-point format is used for all the data.

In the experiments, the reconstruction FOVs are all $[-1, 1]$ in x, y, and z axes. If 512 slices are reconstructed, the thickness of a single slice is $dz = 2/512$. Number of slices allocated on device memory is

$$\text{Slice}_{\text{GPU}} = \frac{2D_{\text{overscan}}}{dz} = \frac{2 \times 0.3588p}{dz} = 129.$$

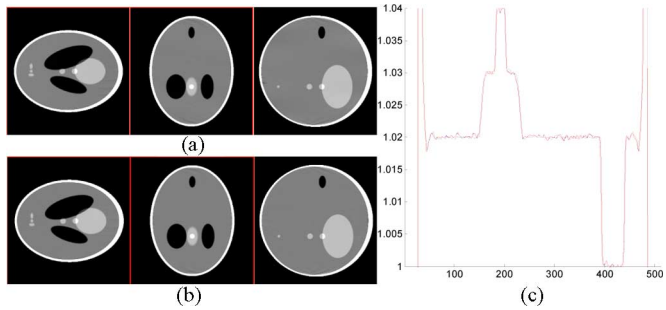


Fig. 9. Reconstruction images in (a) flat-detector geometry using GPU and (b) curved-detector geometry using GPU. (c) The profiles along an axis parallel to the z-axis at $x = 0.1230$ and $y = 0.0059$. The red and blue color profiles are from flat-detector geometry using GPU and CPU, respectively, and the dark color profiles is from curved-detector geometry using GPU. The density scale window is $[1.00, 1.04]$.

Usually, more slices are allocated on device memory than the theoretical value because a smooth function $\chi(u, v)$, which is larger than Tam–Danielsson window, is used to avoid reconstruction artifacts. Here $\text{Slice}_{\text{GPU}}$ is set to 68, 135, 270 for 256, 512, and 1024 slices, respectively.

Fig. 9 shows the reconstruction results of a 512^3 volume at $z = -0.2480, x = 0.1230$, and $y = 0.0059$, and the density scale window is set to $[-1.00, 1.04]$. Fig. 9(a) and (b) illustrate the reconstruction images in flat-detector geometry and curved-detector geometry using GPU, respectively. Fig. 9(c) shows the profiles along an axis parallel to the z-axis at $x = 0.1230$ and $y = 0.0059$, where the red and blue color profiles are from flat-detector geometry using GPU and CPU, respectively, and the dark color profiles is from curved-detector geometry using GPU.

Table III shows the time consumptions of different volume datasets without blocking and with blocking method in the flat-detector geometry of Table II. Filtering represents the CPU filtering time. BP stands for the backprojection time without blocking method, while $'\_b'$ means timing measurements with blocking method. The CRTT technology [16] is applied when using 3-D texture.

If the reconstructed volume size is less than the device memory, which is 768 MB of the NVIDIA GeForce 8800GTX card, as shown in Table III, the speed of backprojection is proportional to the rate of total slice number and slice number allocated on device memory.

If the reconstructed volume size is larger than the device memory, like a $1024 \times 1024 \times 256$ volume in float type of the size 1024 MB, the 3-D texture would not work without using the blocking method. If a stack of 2-D textures are used for storing the reconstructed volume, the backprojection time is about 262.1 s without using blocking method. In contrast, if the blocking method is used, it takes only 10.9 s to perform the same reconstruction.

As shown in Table III, the summation of the filtering time and backprojection time is larger than the total reconstruction time in some cases, because there is some degree of parallelism of CPU and GPU. While reconstructing large volume without blocking method that needs data transfer between main memory and device memory nearly do nothing in the parallelism of CPU and GPU.

TABLE III
TIME CONSUMPTIONS OF DIFFERENT VOLUME DATASETS USING NONBLOCK METHOD AND BLOCK METHOD IN THE FLAT-DETECTOR GEOMETRY OF TABLE II

Volume	Filtering	BP	Total	Filtering_b	BP_b	Total_b	Hardware
512^3	11.7	18.0	18.2	11.9	5.9	16.5	GPU 2D texture
	11.7	13.4	16.0	11.7	4.1	15.0	GPU 3D texture
$512^2 \times 1024$	11.6	331.1	342.7	11.8	11.5	20.7	GPU 2D texture
	~	~	~	11.7	7.9	17.9	GPU 3D texture
$1024^2 \times 256$	11.7	262.1	273.8	11.8	10.9	19.9	GPU 2D texture
	~	~	~	12.1	7.6	17.3	GPU 3D texture

Filtering represents the CPU filtering time. BP stands for the backprojection time without blocking method. while "_b" means timing measurements using block method. "~" represents of no mesurement. The timing unit is second.

TABLE IV
TIME CONSUMPTIONS OF DIFFERENT VOLUME DATASETS USING NONBLOCK METHOD AND BLOCK METHOD IN THE CURVE GEOMETRY OF TABLE II

Volume	Filtering	BP	Total	Filtering_b	BP_b	Total_b	Hardware
512^3	11.1	33.8	34.0	10.6	9.5	17.1	GPU 3D texture
$512^2 \times 1024$	~	~	~	10.7	18.7	25.2	GPU 3D texture

Filtering represents the CPU filtering time. BP stands for the backprojection time without blocking method. while "_b" means timing measurements using block method. The timing unit is second.

TABLE V
PERFORMANCE COMPARISONS FOR HELICAL CONE-BEAM CT RECONSTRUCTION

	Hardware	GUPS	Note
Deng *et al*. [10]	cluster (32 processors)	~	not enough information
Yang *et al*. [11]	cluster (32 processors)	0.21	
Fontaine and Lee[12]	CPU (8 cores)	~	not enough information
Steckmann *el al*. [24]	CPU (8 cores)	9.7	Using symmetry
Steckmann *el al*. [24]	CPU (24 cores)	17	Using symmetry
This paper	GPU(8800GTX)	4.9	

Goddard *et al.* [23] proposed Giga updates (GUs) per second (GUPS) to measure the reconstruction speed, where GUs is the division of the total number of voxel updates and 1024^3. For helical geometry, there is no easy way to calculate the exact GUs. A rough estimate of the GUs can be done as follows. The backprojection of a single slice needs $2 \times \text{overscan} \times (180/\pi) = 258$ projections. Hence, for reconstruction a 512^3 volume, an upper bound for the GUs is $512^3 \times 258 \times (\pi/4)/1024^3 = 25.3$ GU, the factor $\pi/4$ accounts for the fact that the reconstruction volume contains voxels outside the field of measurement (FOM) that need not to be backprojection. We also count the actually required updates in our algorithm, which is 20.0 GU.

In the flat-detector geometry, our algorithm takes about 4.1 s to perform backprojection of a 512^3 volume from 1288 views with the size 672×128; therefore, the reconstruction speed is 20.0/4.1 = 4.9 GUPS. In the curved-detector geometry, due to more complicated coordinate calculations for the curved detector, the reconstruction performance is lower than the flat-panel detector, as shown in Table IV. As iillustrated in (22) and (23), calculating inverse trigonometric function and trigonometric function is time-consuming, which results in the different performances for the flat detector and the curved detector.

Table V shows comparisons of the reconstruction performance for helical cone-beam CT from different groups. Deng *et al.* [10], Yang *et al.* [11], and Fontaine and Lee [12] focus on the acceleration of Katsevich algorithm. In [10], it takes 185 s to reconstruct a 512^3 volume on a 32 processors cluster, but there is little other reconstruction information; therefore, we can not estimate its performance in terms of GUPS. In [11], for the 3-D Shepp–Logan phantom with 256^3 voxels, 3×600 source points (600 points per turn) and 100×500 cone-beam projection at every source point, the proposed parallel implementation needs 25.7 s with 32 processors on a Linux cluster. According to its geometry information and (24), the overscan is 2.255 radians; therefore, the backprojection of a single slice needs $2 \times 2.255 \times (600/(2\pi)) = 431$ projections. Hence, for reconstruction a 256^3 volume, an upper bound for the GUPS is $256^3 \times 431 \times (\pi/4)/1024^3/25.7 = 0.21$ GUPS. Fontaine and Lee [12] and Steckmann *et al.* [24] use geometry symmetry to accelerate the reconstruction speed by multicore CPU. Fontaine and Lee [12] show that it takes 642 s to reconstruct a 1024^3 image using 5120 projections with 512×128 size on a dual-socket quad-core system. But reconstruction information is not enough, and the GUPS can not be estimated. Steckmann *el al.* [24] propose a new algorithm that utilizes the spiral symmetry. It achieves up to 9.7 GUPS running on a Celsius R650 workstation with eight core processors CPU, and it achieves up to 17 GUPS on their systems that are equipped with four standard Intel X7460 hexa core CPUs (Intel Xeon 7300 platform, 2.66 GHz, Intel Corporation).

V. Conclusion

The modern GPU hardware, with its programmable features and powerful computational performance, is very suitable for computational intensive work. It is shown that GPU is more cost-effective for 3-D CT reconstruction than cluster. In this paper, Katsevich algorithm is accelerated by using GPU hardware, and an overscan formula is derived, which helps to avoid unnecessary overscan in practical CT scanning. Based on the overscan formula, a volume-blocking method in device memory has been proposed, which saves the limited device memory. The blocking method based on overscan formula can solve the conflict between large volume and high speed, effectively, only if the device memory can hold volume for two overscan distance.

References

[1] L. Feldkamp, L. Davis, and J. Kress, "Practical cone-beam algorithm," *J. Opt. Soc. Amer. A, Opt. Image Sci., Vis.*, vol. 1, no. 6, pp. 612–619, 1984.

[2] G. Wang, T. H. Lin, P. C. Cheng, and D. M. Shinozaki, "A general cone-beam reconstruction algorithm," *IEEE Trans. Med. Imag.*, vol. 12, no. 3, pp. 486–496, Sep. 1993.

[3] A. Katsevich, "Theoretically exact filtered backprojection-type inversion algorithm for spiral CT," *SIAM J. Appl. Math.*, vol. 62, no. 6, pp. 2012–2026, 2002.

[4] A. Katsevich, "An improved exact filtered backprojection algorithm for spiral computed tomography," *Adv. Appl. Math.*, vol. 32, no. 4, pp. 681–697, 2004.

[5] Y. Zou and X. C. Pan, "Image reconstruction on PI-lines by use of filtered backprojection in helical cone-beam CT," *Phys. Med. Biol.*, vol. 49, no. 12, pp. 2717–2731, 2004.

[6] Y. Zou and X. C. Pan, "Exact image reconstruction on PI-lines from minimum data in helical cone-beam CT," *Phys. Med. Biol.*, vol. 49, no. 6, pp. 941–959, 2004.

[7] Y. Zou, X. C. Pan, and E. Y. Sidky, "Theory and algorithms for image reconstruction on chords and within regions of interest," *J. Opt. Soc. Amer. A, Opt. Image Sci. Vis.*, vol. 22, no. 11, pp. 2372–2384, 2005.

[8] J. D. Pack, F. Noo, and R. Clackdoyle, "Cone-beam reconstruction using the backprojection of locally filtered projections," *IEEE Trans. Med. Imag.*, vol. 24, no. 1, pp. 70–85, Jan. 2005.

[9] Y. B. Ye, S. Y. Zhao, H. Y. Yu, and G. Wang, "A general exact reconstruction for cone-beam CT via backprojection-filtration," *IEEE Trans. Med. Imag.*, vol. 24, no. 9, pp. 1190–1198, Sep. 2005.

[10] J. J. Deng, H. Y. Yu, J. Ni, T. He, S. Y. Zhao, L. H. Wang, and G. Wang, "A parallel implementation of the katsevich algorithm for 3-D CT image reconstruction," *J. Supercomput.*, vol. 38, no. 1, pp. 35–47, 2006.

[11] J. Yang, X. Guo, Q. Kong, T. Zhou, and M. Jiang, "Parallel implementation of Katsevich's FBP algorithm," *Int. J. Biomed. Imag.*, vol. 2006, pp. 1–8, 2006.

[12] E. Fontaine and H. H. S. Lee, "Optimizing Katsevich image reconstruction algorithm on multicore processors," in *Proc. Int. Conf. Parallel Distrib. Syst.*, Hsinchu, Taiwan: IEEE, 2007, pp. 467–474.

[13] B. Cabral, N. Cam, and J. Foran, "Accelerated volume rendering and tomographic reconstruction using texture mapping hardware," in *Proc. Symp. Vol. Vis.*, Tysons Corner, Virginia: ACM, 1994, pp. 91–98.

[14] F. Xu and K. Mueller, "Real-time 3D computed tomographic reconstruction using commodity graphics hardware," *Phys. Med. Biol.*, vol. 52, no. 12, pp. 3405–3419, 2007.

[15] H. Yang, M. Li, K. Koizumi, and H. Kudo, "Accelerating backprojections via CUDA architecture," in *Proc. 9th Int. Meet. Fully Three-Dimensional Image Reconstr. Radiol. Nuclear Med.*, Lindau, Germany, 2007, pp. 52–55.

[16] G. R. Yan, J. Tian, S. P. Zhu, Y. K. Dai, and C. H. Qin, "Fast cone-beam CT image reconstruction using GPU hardware," *J. X-Ray Sci. Technol.*, vol. 16, no. 4, pp. 225–234, 2008.

[17] F. Noo, J. Pack, and D. Heuscher, "Exact helical reconstruction using native cone-beam geometries," *Phys. Med. Biol.*, vol. 48, no. 23, pp. 3787–3818, 2003.

[18] P.-E. Danielsson, P. Edholm, J. Eriksson, and M. Magnusson-Seger, "Towards exact 3-D reconstruction for helical cone-beam scanning of long objects: A new arrangement and a new completeness condition," in *Proc. Int. Meeting Fully Three-Dimensional Image Reconstr. Radiol. Nuclear Med.*, Nemacolin, PA, 1997, pp. 141–144.

[19] M. Defrise, F. Noo, and H. Kudo, "A solution to the long-object problem in helical cone-beam tomography," *Phys. Med. Biol.*, vol. 45, no. 3, pp. 623–643, 2000.

[20] K. C. Tam, S. Samarasekera, and F. Sauer, "Exact cone beam CT with a spiral scan," *Phys. Med. Biol.*, vol. 43, no. 4, pp. 1015–1024, 1998.

[21] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proc. IEEE*, vol. 93, no. 2, pp. 216–231, Feb. 2005.

[22] M. Segal, C. Korobkin, R. v. Widenfelt, J. Foran, and P. Haeberli, "Fast shadows and lighting effects using texture mapping," in *Proc. 19th Annu. Conf. Comput. Graph. Interactive Tech.*, New York: ACM, 1992, pp. 249–252.

[23] I. Goddard, A. Berman, O. Bockenbach, F. Lauginiger, S. Schuberth, and S. Thieret, "Evolution of computer technology for fast cone beam backprojection," presented at the SPIE Comput. Imag. V, San Jose, CA, 2007.

[24] S. Steckmann, M. Knaup, and M. Kachelriess, "High performance cone-beam spiral backprojection with voxel-specific weighting," *Phys. Med. Biol.*, vol. 54, no. 12, pp. 3691–3708, 2008.

Guorui Yan received the B.S. degree in information engineering from Xi'an Jiaotong University, Xi'an, China, in 2006.

He is currently a Team Member of Medical Imaging ToolKit (MITK) in the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, Beijing, China. Since 2006, he has engaged in the development of MITK and 3-D medical image processing and analyzing system since 2006. His research interests include computed tomography reconstruction and graphics processing unit acceleration of general purpose computing.

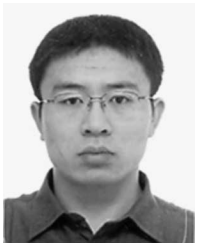
Jie Tian (M'03–SM'03–F'10) received the Ph.D. degree (with honor) in artificial intelligence from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1992.

From 1995 to 1996, he was a Postdoctoral Fellow with the Medical Image Processing Group, University of Pennsylvania, Philadelphia. He is also with the Life Science Center, Xidian University, Xi'an. Since 1997, he has been a Professor with the Medical Image Processing Group, Institute of Automation. His current research interests include medical image process and analysis, pattern recognition. He has authored or coauthored more than 70 research papers in the international journals and conferences.

Dr. Tian is the Beijing Chapter Chair of the engineering in Medicine and Biology Society of the IEEE.

Shouping Zhu received the B.S. degree in biomedical engineering and the M.S. degree in signal and information processing from Xidian University, Xi'an, China, in 2004 and 2007, respectively. He is currently working toward the Ph.D. degree with the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, Beijing, China

His research interests include medical image processing, microcomputed tomography (CT) system, and CT reconstruction.



Chenghu Qin (M'09) received the M.S. degree in signal and information processing from Tianjin University, Tianjin, China, in 2006, and the Ph.D. degree in computer application technology from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009.

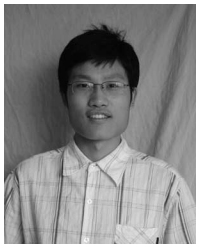
He is currently an Assistant Researcher with the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include medical image processing and optical molecular imaging.

Yakang Dai received the B.S. degree in electrical engineering from Hunan University, Changsha, China, in 2004.

He is currently a Member with the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, Beijing, China. Since 2004, he has been engaged in the development of Medical Imaging Toolkit and 3-D medical image processing and analyzing system. His research interests include 3-D imaging and visualization.

Fei Yang received the B.S. degree in biomedical engineering from Beijing Jiaotong University, China, in 2009. He is currently working toward the Ph.D. degree in computer science with the Institute of Automation, Chinese Academy of Sciences, China.

Since 2008, he has been engaged in the development of Medical Imaging ToolKit and 3-D medical image processing and analyzing system. His current research interests include graphics processing unit - based visualization and scientific computing.



Di Dong received the B.S. degree in automation from the University of Science and Technology Beijing, Beijing, China, in 2008.

He is currently a Member with the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, Beijing. Since 2008, he has been engaged in the development of Medical Imaging Toolkit and 3-D medical image processing and analyzing system. His current research interests include freehand ultrasound imaging and computed tomography imaging.

Ping Wu received the B.S. degree in electrical engineering from the Central South University, Changsha, China, in 2009. She is currently working toward the Ph.D. degree in computer science with the Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, Beijing, China.

Since 2009, she has been engaged in the development of Medical Imaging Toolkit and 3-D medical image processing and analyzing system. Her current research interest include computed tomography reconstruction.