Perspective Projection through Parallelly Projected Slabs for Virtual Endoscopy

Anna Vilanova* Rainer Wegenkittl† Andreas König *† Eduard Gröller*

* Institute of Computer Graphics
Vienna University of Technology

† Tiani Medgraph

Abstract

Virtual Endoscopy is a promising medical application for volume rendering techniques where perspective projection is mandatory. Most of the acceleration techniques for direct volume rendering make use of parallel projection. This is also the case of the current generation of VolumePro systems, which achieve real-time frame rates but unfortunately just provide parallel projection.

In this paper, an algorithm to approximate perspective volume rendering using parallelly projected slabs is presented. The introduced error due to the approximation is investigated. Based on the error estimation, an improvement to the basic algorithm is presented. The improvement increases the frame rate keeping the global maximal error bounded.

The usability of the algorithm is shown through the virtual endoscopic investigation of various types of medical data sets.

Keywords: Direct Volume Rendering, Virtual Endoscopy, Perspective Projection, VolumePro Technology

1 Introduction

The visualization of medical volume data produced by 3D imaging techniques (e.g. CT and MRI) has been intensively investigated in the last decades. Its application to daily medical care can highly improve the quality of current medical procedures.

Virtual endoscopy is an application which deals with the exploration of hollow organs and anatomical cavities using volume data. Virtual endoscopy has the potential of being used as a non-invasive diagnostic technique. It may also be applied in surgical planning, training and exploration of structures not reachable with a real endoscope. All these fields have similar requirements concerning the visualization system : accuracy, intuitive interaction, fast visualization, and short preprocessing.

Several virtual endoscopy systems have been proposed [5, 11, 12]. These systems are basically concerned with two visualization techniques: surface rendering and direct volume rendering. Surface rendering leads to a reduction of the data information from 3D (volume) to 2D (surfaces). This incurs a loss of information and accuracy while requiring surface extraction as a preprocessing step. On the other hand, the advantage is that common graphics hardware can be used to accelerate the rendering step.

Direct volume rendering uses the 3D information and projects it to a 2D image plane (e.g. with ray-casting [7]). It has the disadvantage that these projection algorithms are computationally rather expensive.

*email:{anna,koenig,groeller}@cg.tuwien.ac.at, http//:www.cg.tuwien.ac.at

†email:{rainer.wegenkittl}@tiani.com, http//:www.tiani.com

This paper concentrates on direct volume rendering techniques since no preprocessing is necessary and higher accuracy is achieved.

There are several hardware [3, 9, 10] and software [1, 4, 6, 13, 14] acceleration techniques to improve the frame rate of direct volume rendering. The software accelerated techniques usually need additional data storage and preprocessing.

3D texture mapping [2, 3] is the most often used hardware acceleration technique. This method achieves interactive frame rates on an SGI Reality Engine, but it is difficult to incorporate this technique into a desktop machine like a PC. The basic method does not support the possibility to estimate gradients which is required to employ lighting models like the Phong model with diffuse and specular lighting effects. However several approaches to overcome this problem have been proposed [15].

Recently, the VolumePro board [9] was released. It is a hardware implementation of ray-casting using shear-warp factorization [6]. It provides real-time rendering with compositing [8], classification with density based transfer functions and Phong shading.

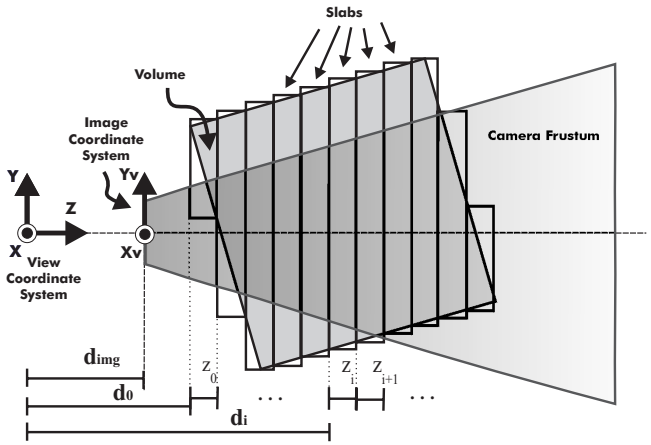


Figure 1: Illustration of the perspective approximation using the projected-slabs algorithm.

One of the main drawbacks of this board, with regard to usage in virtual endoscopy, is that it does not produce perspective projection. For outside views, parallel projection gives good results but for inside views (e.g. endoscopic views), perspective projection is mandatory to provide a correct depth impression.

SpaceVision, which is developed by Tiani MedGraph in cooperation with the Vienna University of Technology, is a 3D medical visualization package already in use. VolumePro technology is used in SpaceVision to achieve volume rendering at interactive frame-rates. The work presented in this paper will be included in

SpaceVision as a rendering technique for virtual endoscopy applications.

In this paper, a method to approximate perspective projection from several parallelly projected slabs, similar to slicing [3], is presented. An error estimation of the approximation is studied and an improvement in the performance of the initial algorithm by using the error estimation is described. Some problems are investigated and improvements are presented. Finally, a study using several clinical data sets and performance issues are discussed.

Actually, the presented algorithm is not only restricted to the VolumePro application. More generally, the concept can be applied wherever perspective projection is used.

2 Projected-Slabs Algorithm

The VolumePro system is able to produce high quality volume rendering of about 30 frames per second for a 256 cubic-size volume-data.

The VolumePro technology implements a shear-warp algorithm [6] in hardware. It renders the baseplane image and the 2D warp operation must be done by common graphics hardware.

The basic idea of the presented algorithm, called projected-slabs algorithm, is as follows: generate perspective rendering of the entire 3D data set, approximated by consecutive parallel projections of slabs of the volume data (see figure 1). A slab is part of the volume data in between two cutting planes which are orthogonal to the viewing direction. The thickness of a slab is selected such that the difference between a parallel projection and a perspective projection of the volume data contained within a slab is tolerable (i.e. below a certain error threshold). Using the cutting plane feature of the VolumePro system, each slab is rendered using parallel projection. The resulting baseplane image of an individual slab is then warped and transformed according to the perspective parameters of the defined camera.

All the images, one per slab, are finally blended to get the image of the entire data set. Figure 2 illustrates which data values are accumulated along a viewing ray with the projected-slabs methods compared to the correct perspective solution.

This algorithm uses an entire VolumePro rendering cycle for every rendered slab and therefore the rendering frame rate is decreasing in the order of the number of slabs that are needed for the perspective approximation. Besides, there is an overhead due to the blending of the slabs.

Given the view position and the viewing direction, the slabs can be numbered using the distance to the viewpoint in the following way (see figure 1):

$$d_j = d_0 + \sum_{i=0}^{j-1} \Delta_{Z_i} \tag{1}$$

where $j \geq 1$ and d_0 is the distance from the viewpoint to the front plane of the first slab, and Δ_{Z_i} is the thickness of the slab which starts at distance d_i.

If Δ_{Z_i} is a constant value smaller or equal to a voxel size for all the slabs, it is intuitive to see that the result produces good quality perspective rendering. On the other hand, it also produces an intolerable high number of slabs. So the thickness of the slabs must be set to a value larger than one voxel size to get reasonable performance.

Since we are approximating perspective projection, it is important to be able to evaluate the error produced due to this approximation.

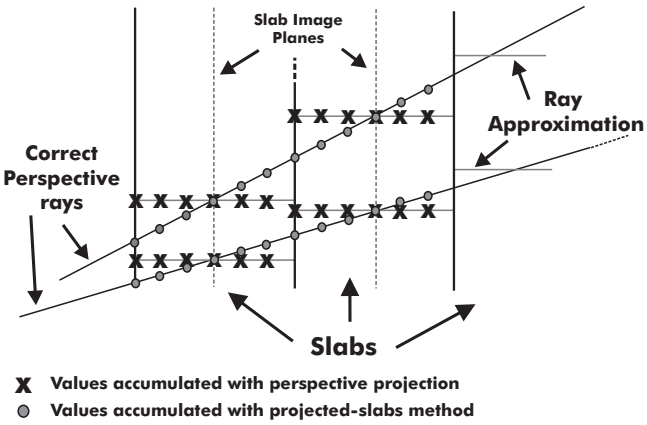


Figure 2: Accumulated values in a correct perspective projection as compared to the projected-slabs algorithm

3 Error Estimation of the Projected-Slabs Algorithm

In this section, we study the error that results from the use of parallelly projected slabs to produce the perspective view.

The basic error is that the sample points are projected to the wrong position in the image plane, and therefore they are accumulated to the wrong ray. Based on that, the error estimation is defined as the distance in the image plane between the correct perspectively projected point and the point produced by the projected-slabs algorithm.

In the rendering pipeline, the difference between parallel and perspective projection appears after the world coordinates have been transformed to view coordinates. To transform from view coordinates to image plane coordinates, the appropriate projection matrix is used. For simplicity and more intuitive explanation, we assume a left handed camera system (see figure 1) where the viewpoint is in the origin of the view coordinates. The image plane is orthogonal to the Z-direction and located at a distance d_{img} from the viewpoint.

We define a point $P_v = (X_v, Y_v, Z_v)$ as a point resulting by applying a view-coordinate transformation to an arbitrary point in world-coordinates.

The perspective projection matrix for a left handed camera system and supposing left accumulation matrix notation is:

$$M_{persp} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \frac{1}{d} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where d is the distance from the viewpoint to an arbitrary projection plane. If $d = d_{img}$ then the projection plane is the image plane. The parallel transformation to image coordinates is simply the transformation of the Z-coordinate to the projection plane position d. It can be expressed by the matrix:

$$M_{paral} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d & 1 \end{bmatrix}$$

If these transformations are applied to P_v we obtain the following equalities (P^h are points expressed in homogeneous coordinates).

$$P^h_{persp} \quad = \quad P^h_v * M_{persp} = \left[X_v, Y_v, Z_v, \frac{Z_v}{d} \right]$$

$$P^h_{paral} = P^h_v * M_{paral} = [X_v, Y_v, d, 1]$$

The points P^h_{persp} and P^h_{paral} are transformed to the 2D projection plane coordinate system. This coordinate system is defined with the same X and Y-direction as the view coordinate system. The projection plane origin corresponds to $[0,0,d]$ in view-coordinates. Then P^h_{persp} corresponds to P_{persp} and P^h_{paral} corresponds to P_{paral}, with:

$$\begin{aligned} P_{persp} &= \frac{d}{Z_v}[X_v, Y_v] \\ P_{paral} &= [X_v, Y_v] \end{aligned}$$

Error e_p is defined as the distance between the perspective, P_{persp}, and parallel, P_{paral}, projection of a point P_v:

$$\begin{aligned} e_p &= \|P_{persp} - P_{paral}\| = \\ &= \left\| \left(\frac{d}{Z_v} - 1 \right) [X_v, Y_v] \right\| \end{aligned}$$

Z_v can be expressed by $Z_v = d + \Delta_{Z_v}$. Then we derive:

$$\begin{aligned} e_p &= \left\| \left(\frac{d}{d+\Delta_{Z_v}} - 1 \right) [X_v, Y_v] \right\| = \\ &= \left| \left(\frac{\Delta_{Z_v}}{d+\Delta_{Z_v}} \right) \right| \| [X_v, Y_v] \| \end{aligned} \quad (2)$$

where $-\infty \leq \Delta_{Z_v} \leq \infty$

Equation 2 represents the distance between a parallel projection and perspective projection of a point on a projection plane situated at a distance d from the viewpoint.

In the projected-slabs algorithm, for each slab i the points within the slab are parallelly projected to the slab image plane. The slab image plane is situated in the middle of the slab, at distance $d_i + \frac{\Delta_{Z_i}}{2}$ from the viewpoint (see figure 3).

We define e_{s_i} as the distance between the parallel projection and the perspective projection of a point on the slab image plane. Using equation 2 and $d = d_i + \frac{\Delta_{Z_i}}{2}$, it follows:

$$e_{s_i} = \left| \left(\frac{\Delta_{Z_v}}{d_i + \frac{\Delta_{Z_i}}{2} + \Delta_{Z_v}} \right) \right| \| [X_v, Y_v] \| \quad (3)$$

where $-\frac{\Delta_{Z_i}}{2} \leq \Delta_{Z_v} < \frac{\Delta_{Z_i}}{2}$.

Equation 3 gives a distance in the slab image plane. However, we are interested in this distance projected into the image plane. Therefore, we project e_{s_i} to the image plane and get e_{img_i}:

$$e_{img_i} = \left| \left(\frac{\Delta_{Z_v}}{d_i + \frac{\Delta_{Z_i}}{2} + \Delta_{Z_v}} \right) \right| \left(\frac{d_{img}}{d_i + \frac{\Delta_{Z_i}}{2}} \right) \| [X_v, Y_v] \| \quad (4)$$

where d_{img} represents the distance between the image plane and the viewpoint.

The point P_v can be expressed as follows:

$$P_v = \frac{Z_v}{d_{img}} * [X_p, Y_p, d_{img}] \quad (5)$$

where $P_p = [X_p, Y_p, d_{img}]$ is the perspective projection of the point P_v on the image plane.

Combining equation 4 together with 5 it follows:

$$\begin{aligned} e_{img_i} = & \left| \left(\frac{\Delta_{Z_v}}{d_i + \frac{\Delta_{Z_i}}{2} + \Delta_{Z_v}} \right) \right| \left(\frac{d_{img}}{d_i + \frac{\Delta_{Z_i}}{2}} \right) \\ & \left| \left(\frac{d_i + \frac{\Delta_{Z_i}}{2} + \Delta_{Z_v}}{d_{img}} \right) \right| \| \left[X_p, Y_p \right] \| \end{aligned}$$

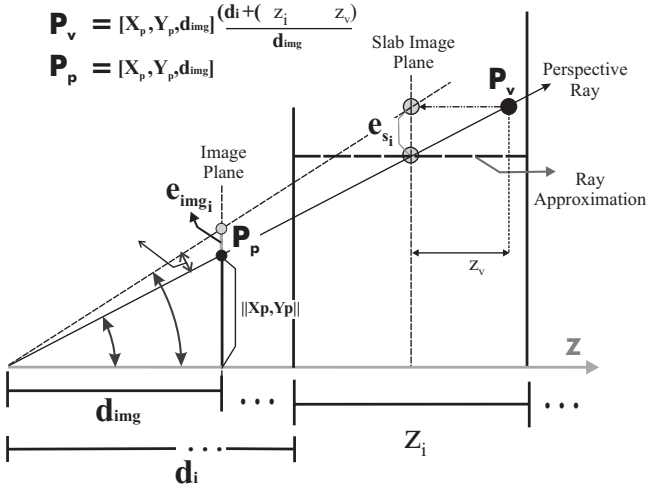


Figure 3: Illustration of the error estimation.

where $-\frac{\Delta_{Z_i}}{2} \leq \Delta_{Z_v} < \frac{\Delta_{Z_i}}{2}$.

Simplifying the previous equation it follows:

$$e_{img_i} = \left(\frac{\Delta_{Z_v}}{d_i + \frac{\Delta_{Z_i}}{2}} \right) \| \left[X_p, Y_p \right] \| \quad (6)$$

where $0 \leq \Delta_{Z_v} < \frac{\Delta_{Z_i}}{2}$.

From equation 6, we can deduce several properties:

1. The error e_{img_i} is proportional to $\| \left[X_p, Y_p \right] \|$, the distance of P_p to the image plane origin.
2. $\forall [X_p, Y_p]$: $\Delta_{Z_v} \to 0 \Rightarrow e_{img_i} \to 0$. This behaviour has already been intuitively described in section 2. Besides, it means that for points on the slab image plane the error is 0.
3. The error value increases, if Δ_{Z_v} increases.
4. For fixed $\| \left[X_p, Y_p \right] \|$, if d_i increases, then the variation in e_{img_i} due to the changes in Δ_{Z_v} decreases.

The value of e_{img_i} represents the error or distance in the image plane between the projected-slabs algorithm and the correct perspective projection of a point P_v situated in the slab i. The point is situated at a distance Δ_{Z_v} from the slab image plane and its perspective projection to the image plane gives $[X_p, Y_p]$.

We are interested in finding the maximum error produced in the image plane due to the approximation of the projected-slabs algorithm. This can be achieved by finding the upper bound for all values of e_{img_i}.

Due to property 3, it is clear that e_{img_i} is maximal when $\Delta_{Z_v} = \frac{\Delta_{Z_i}}{2}$ i.e. when the point P_v is situated on the front or back plane of the slab.

Due to property 1, it is clear that the upper bound value of e_{img_i} within slab i occurs when the highest value of $\| \left[X_p, Y_p \right] \|$ is reached. That implies that $[X_p, Y_p]$ can be fixed to the farthest point from the origin of the image plane that contributes to the final image. This value corresponds the corners of the image quadrilateral defined by the intersection of the frustum and the image plane (i.e. $[X_c, Y_c]$). Therefore we define e^{max}_i as the maximum error in the final image inferred by slab i.

$$e^{max}_i = \left(\frac{\frac{\Delta_{Z_i}}{2}}{d_i + \frac{\Delta_{Z_i}}{2}} \right) \| [X_c, Y_c] \| \quad (7)$$

The maximal error in the final image is the maximum value of e_i^{max} for any slab i.

$$e^{max} = max\{e_i^{max} | i \geq 0\} \quad (8)$$

Due to property 4 we see that if for all i, Δ_{Z_i} is a constant , then every slab has a different e_i^{max} and its value decreases when d_i increases. So, the maximal error produced in the final blended image corresponds to the maximal error of the first slab, $e^{max} = e_0^{max}$.

In the next section we use the previous observations to optimize the projected-slabs algorithm, without increasing the maximal error value e^{max} produced in the final image.

4 Error-Induced Variation of Slab Thickness

In the previous section it has been observed that the maximal error for slab i due to the projected-slabs algorithm, e_i^{max}, depends on the distance of the slab to the viewpoint and on the slab thickness. From property 4 and equation 8, it can be deduced that slabs further away from the viewpoint may have a greater thickness than slabs closer to the viewpoint keeping the same maximal image error e^{max}.

In this section we present the criterion for selecting the slab thickness dependent on the distance to the viewpoint, the camera characteristics and the maximal error. The rule is to have as few slabs as possible while keeping the error tolerance unchanged.

Using equation 7, Δ_{Z_i} can be isolated in the following way:

$$\Delta_{Z_i} = 2 * \left(\frac{e_i^{max} * d_i}{|| [X_c, Y_c] || - e_i^{max}} \right) \quad (9)$$

The e_i^{max} is set to a constant value *DistanceError*, for all the slabs. We defined the incremental slab thickness algorithm using equations 9 and 1. The thickness of the slabs is defined in an iterative way, assuring that the error will be kept smaller than the defined value *DistanceError*. As was mentioned in section 2 the projected-slabs algorithm decreases the frame rate if the number of slabs increases. Calculating the thickness using equation 9, the slab thickness will increase with the value of d_i and so, less slabs are needed and therefore the frame rate is higher (see figure 4).

Apart from the *DistanceError* the equation 9 also depends on the camera characteristics: $[X_c, Y_c]$ is specified by the intersection of the frustum and the image plane. Δ_{Z_i} just needs to be computed when the camera characteristics, the *DistanceError* or the first slab distance d_0 are modified.

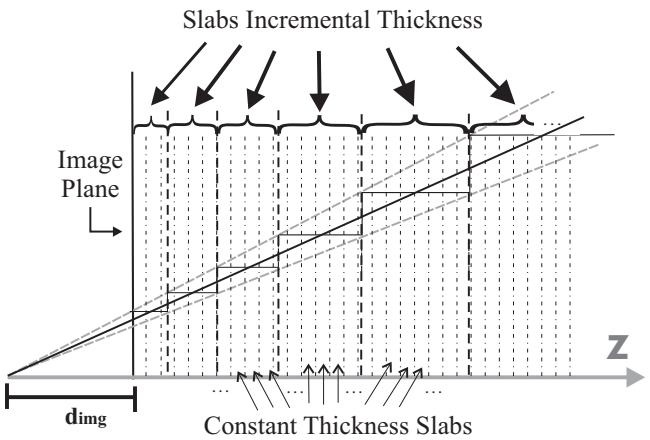


Figure 4: Comparison of the slabs using constant thickness and the incremental slab thickness calculation

Observing figure 3 and using simple trigonometry, Δ_{Z_i} can be intuitively described as a function of the angles α and β:

$$\tan\beta * \left(d_i + \frac{\Delta_{Z_i}}{2} \right) = \tan\alpha * \left(d_i + \frac{\Delta_{Z_i}}{2} + \Delta_{Z_v} \right)$$

where $-\frac{\Delta_{Z_i}}{2} \leq \Delta_{Z_v} < \frac{\Delta_{Z_i}}{2}$ and $\beta = \alpha + \Delta_\alpha$. Δ_α corresponds to the angle defined between the ray where a point is accumulated using correct perspective projection and the ray that accumulates the point in the projected-slabs method. We can observe that the maximum value of Δ_α within a slab is reached when $\Delta_{Z_v} = \pm\frac{\Delta_{Z_i}}{2}$. For simplicity we assumed $\Delta_{Z_v} = -\frac{\Delta_{Z_i}}{2}$. It follows:

$$\Delta_{Z_i} = 2 * \left(\frac{\tan\alpha * d_i}{\tan\beta} - d_i \right) \quad (10)$$

Equation 10 gives the option to express the error tolerance as an angle, *AngleError*. It can be seen that α in equation 10 must be set to the maximum frustum angle of the camera to get the maximal value of Δ_α. With some simple transformations, it can be proven that equation 9 and 10 are equivalent.

DistanceError or *AngleError* are parameters of the algorithm. The estimation of the error tolerance (i.e. *DistanceError* or *AngleError*) depends on the volume data to visualize.

To illustrate this, we study one of the worst cases for the projected-slabs algorithm. This case occurs when the camera is in the center of a straight tube and the camera is pointing in the direction of the tube axis.

In parallel projection, a ring with the tube thickness would be projected to the image plane. If the projected-slabs algorithm is used, a ring with the tube thickness for any slab will be projected. If the tube has a small thickness (see figure 5 a), it looks like several concentric rings, giving the impression of having different objects. This is because the specified error tolerance is larger than the projected thickness of the tube. Therefore we see the tube as a discontinuous surface. If the error is decreased to a value that approximates the projection of the thickness in the image plane, a better result is obtained (see figure 5 b).

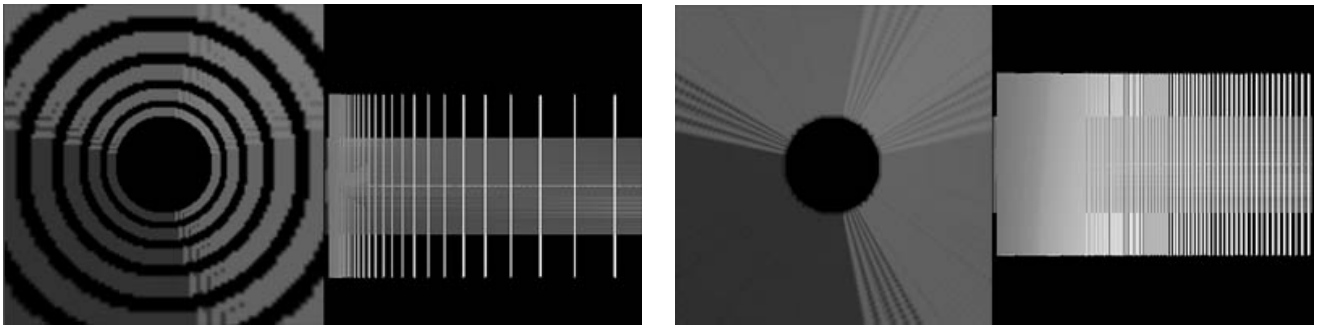
5 Performance Improvements

In this section, we describe several implementation strategies to improve the performance of the algorithm for virtual endoscopy.

Using the projected-slabs algorithm, the possibility to use perspective volume rendering with the VolumePro board is achieved. However, also several problems arise.

One of the problems is that the frame rate does not allow interactive use (i.e. less than 1 f.p.s.) when the number of slabs needed to cover the volume is bigger than 20 or 30. We implemented a progressive rendering algorithm. In this algorithm, the image shown to the user is updated after the rendering of each slab and not just after all the slabs have been processed. In this way the user can see the progressive blending of the slabs and get immediate feedback of the visualization.

There are two ways of compositing in volume rendering: front-to-back and back-to-front. These compositing techniques differ in the processing order of the slabs and the accumulation function applied. In OpenGL, back-to-front compositing is easy to implement, but it implies that the first slabs to be blended are the ones that are farthest from the viewpoint. Usually in virtual endoscopy, the region to explore is situated near the camera. In most of the data visualizations the slabs further from the viewpoint add not much information to the final image. This is because the rays have accumulated completely opaque values before reaching these slabs. Therefore front-to-back compositing has been implemented. The front-to-back compositing order using OpenGL implies a slow down of



a) b)

Figure 5: An illustration of the error tolerance behaviour. Two endoscopic views using the projected-slabs algorithm of a voxelized tube are shown together with the corresponding outside view with the slab image planes. The error tolerance is different for each endoscopic view: *a*) $DistanceError = 5\%$ of the image size, *b*) $DistanceError = 2\%$ of the image size.

the algorithm, since each slab baseplane texture must be multiplied by its own opacity to be able to obtain the correct front-to-back compositing. It produces a reduction of speed of about 30%. We believe this loss is worthwhile since in the incremental front-to-back rendering the interesting regions are visualized first and therefore the user receives faster feedback of his modifications of the rendering parameters.

Another problem to deal with are the aliasing artifacts due to the undersampling of the baseplane image. One option to overcome this problem is to use supersampling. The VolumePro board provides several levels of supersampling. The disadvantage is that supersampling implies a corresponding slow down in rendering performance. The supersampling can be implemented in such a way that the slabs closer to the viewpoint are rendered with a higher supersampling level.

One of the drawbacks of the VolumePro is that only directional light sources can be defined. In virtual endoscopy it is quite convenient to use a head-mounted point light source. In the implementation we use a directional light pointing in the same direction as the camera and two additional directional lights rotated several degrees from the camera direction.

We observed that for wide tubular structures, the slabs nearer to the viewpoint do not contribute to the final image, since the frustum just intersects empty space. This depends, of course, on the data to be visualized. To attenuate this effect, front clipping has been implemented. The user defines the distance from the viewpoint to the first slab.

6 Results

In this section, we present some timings and the evaluation realized for three types of data sets. The calculation times were obtained by executing the algorithm on a Pentium II with 400MHz. The performance of the algorithm presented in this article basically depends on the time necessary for the VolumePro board to render a slab, and on the time for warping and blending. We provide timings relative to the times that can be achieved on a rendering cycle. A rendering cycle consists of the rendering of one slab using the VolumPro board and the warping of the resulting baseplane image. The speed and quality of the warping and blending steps depend on the graphics hardware used.

In tables 1, 2 and 3, the first column denotes the maximal error expressed as a percentage of the image size, the second column gives the number of slabs used, the third column represents the frame rate and the last column denotes the slow down compared with the time of one rendering cycle.

The first dataset is a CT scan of a trachea of size 292x136x114. The results of this visualization with different maximal errors are shown in figure 6. The CT was acquired from a corpse. After the scanning, a real bronchoscopy was performed. Figure 6 compares the results of the projected-slabs approach with the real endoscopic view from a similar camera position. It can be observed that the difference between incremental slab thickness (figure 6a) and the constant slab thickness (figure 6b) can be neglected. Numerical results are given in table 1. The second data set is a CT volume

error	# slabs	f.p.s.	slow down factor
2.5%	37	0.4	40.32
0%	162	0.093	173.40

Table 1: Times for the CT trachea of the corpse, in back-to-front renderings. One rendering cycle takes 65 ms. The error in the second row is 0% since a constant slab thickness of 1 voxels size per slab has been defined.

data of a trachea with a resolution of 205x83x105 (see figure 7). The patient was injected contrast media into the vessels. There is a common surgical procedure (i.e. trans-bronchial biopsy) where the endoscopist does a biopsy of a tumor near the outside walls of the trachea. Using an endoscope, the surgeon penetrates the walls of the trachea from inside at the appropriate position. It is important for the physician to be able to localize the pulmonary artery and the aorta together with the other vessels near the trachea walls. With our system, the endoscopist can navigate until the region of interest is reached. Thereafter the vessels outside the trachea are inspected by rendering the trachea surface transparently. Figure 7 contains the images produced with different opacities for the walls of the trachea. The times are presented in table 2. It can be observed that it is necessary to change the opacity of the trachea walls interactively, since it is difficult to recognize the structure of the trachea when it is semitransparent.

error	# slabs	f.p.s.	slow down factor
4%	21	0.94	22.6

Table 2: CT of a trachea using transparancy, in front-to-back rendering. One rendering cycle takes 47 ms.

The third data set of size 198x115x100 is a portion of a Spiral CT of a colon. In figure 8, a comparison between the projected-slabs technique and brute force volume rendering is presented. For the projected-slabs technique results are presented for both a constant

and an incremental slab thickness. Comparing constant slab thickness with incremental slab thickness, the number of slabs is smaller but the maximum error is the same.

The times and frame rates are given in table 3. We have observed that structures which are wide cavities produce good results since the firstly projected voxels fill a smaller image plane area and aliasing affects the performance of the algorithm less.

error	# slabs	f.p.s.	slow down factor
2.5%	28	0.8	26.58
2.5%	179	0.13	163.61

Table 3: Spiral CT colon data set: Times for a complete back-to-front rendering. One rendering cycle takes 47 ms (21.27 f.p.s.).

We have experienced that around 30 slabs are sufficient for the visualizations.

As has been mentioned in section 4 the projected-slabs algorithm is an approximation which depends on the characteristics of the structure to visualize. This incurs that depending of the structure and the parameters of visualization some artifacts appear. For instance the transition between planes can be observed in some cases where the visualization parameters are not adequate.

For some animations showing more results of the algorithm see `http://www.cg.tuwien.ac.at/research/vis/vismed/projected-slabs/animation.html`.

7 Conclusions and Future Work

An approach to produce perspective projection views using parallel volume rendering techniques (i.e. projected-slabs algorithm) has been presented. The algorithm uses consecutive parallelly projected slabs of the volume. The error produced due to the approximation of perspective projection is investigated. Besides, based on the error studies, we introduce a criterion to vary the slabs thickness and therefore to improve the algorithm performance. The usability of the algorithm has been tested using perspective views for virtual endoscopy in a common desktop machine using the VolumePro system.

As future work, we will speed up the algorithm using active subvolumes. In the projected-slabs algorithm, the entire volume is rendered for every slab but just a small portion of it contributes to the final image. VolumePro allows the definition of an active subvolume with a size equal or less to the original volume size. Instead of the entire volume, the active subvolume is then rendered. The smaller the active subvolume is, the faster the rendering with the VolumePro is. The active subvolume can be defined by the cells of a regular grid that are within the camera frustum and contribute to the rendered slab.

An investigation of how to automate the estimation of the parameters of the algorithm (i.e. front clipping distance and the error tolerance) is another topic for future work.

A study of other uses of the presented error estimation algorithm should be performed. More generally, the algorithm subdivides view space into slabs, within the slabs approximative but faster operations are possible. In our case, parallel projection is used to approximate the perspective view. The concept could also be applied to other algorithms where perspective projection is used (e.g. splatting and shear-warp).

Acknowledgements

The work presented in this publication has been funded by the $V^{is}M^{ed}$ project. $V^{is}M^{ed}$ is supported by *Tiani Medgraph*, Vienna (`http://www.tiani.com`), and the *Forschungsförderungs fonds für die gewerbliche Wirtschaft*, Austria.

See `http://www.vismed.at` for further information on this project.

We thank Dr. Martin C. Freund and the Department of Radiology at Leopold-Franzens-University of Innsbruck for their collaboration and for providing the data used in this paper.

Finally we thank to the Real Time Visualization group `http://www.rtviz.com/`.

References

[1] M. L. Brady, K. K. Jung, H. T. Nguyen, and T. PQ. Nguyen. Interactive volume navigation. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):243–255, July – September 1998.

[2] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *1994 Symposium on Volume Visualization*, pages 91–98. ACM SIGGRAPH, October 1994.

[3] T.J. Cullip and U. Neumann. Accelerating volume reconstruction with 3D texture hardware. Technical Report TR93-027, Department of Computer Science at the University of North Carolina, Chapel Hill, 1993.

[4] J. Danskin and P. Hanrahan. Fast algorithms for volume ray tracing. In *Proceedings of the Workshop on Volume Visualization*, pages 91–98, New York, October19–20 1992. ACM Press.

[5] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He. Virtual voyage: Interactive navigation in the human colon. In *SIGGRAPH 97 Conference Proceedings*, Annual Conference series, pages 27–34. ACM SIGGRAPH, Addison Wesley, August 1997.

[6] P. Lacroute and M. Levoy. Fast volume rendering using a shear–warp factorization of the viewing transformation. In *SIGGRAPH' 94 Conference Proceedings*, pages 451–458. ACM SIGGRAPH, ACM Press, July 1994.

[7] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, February 1987.

[8] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[9] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The VolumePro real-time ray-casting system. In *ACM Siggraph'99*, pages 251–260, 1999.

[10] H. Pfister and A. Kaufman. Cube-4 - A scalable architecture for real-time volume rendering. In *1996 Volume Visualization Symposium*, pages 47–54. IEEE, 1996.

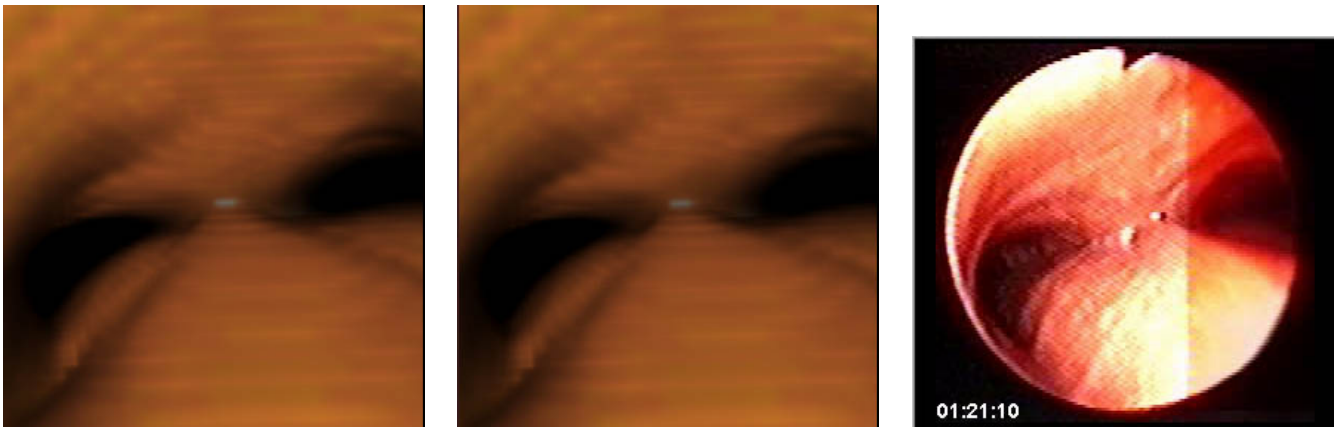
[11] R.M. Satava and R.A. Robb. Virtual endoscopy: Application of 3D visualization to medical diagnosis. *Presence*, 6(2):179–197, April 1997.

[12] A. Vilanova, A. König, and E. Gröller. VirEn: A virtual endoscopy system. *Machine GRAPHICS & VISION*, 8(3):469–487, 1999.

[13] M. Wan, A. Kaufman, and S. Bryson. High performance presence-accelerated ray casting. In *IEEE Visualization '99*, pages 379–386. IEEE, nov 1999.

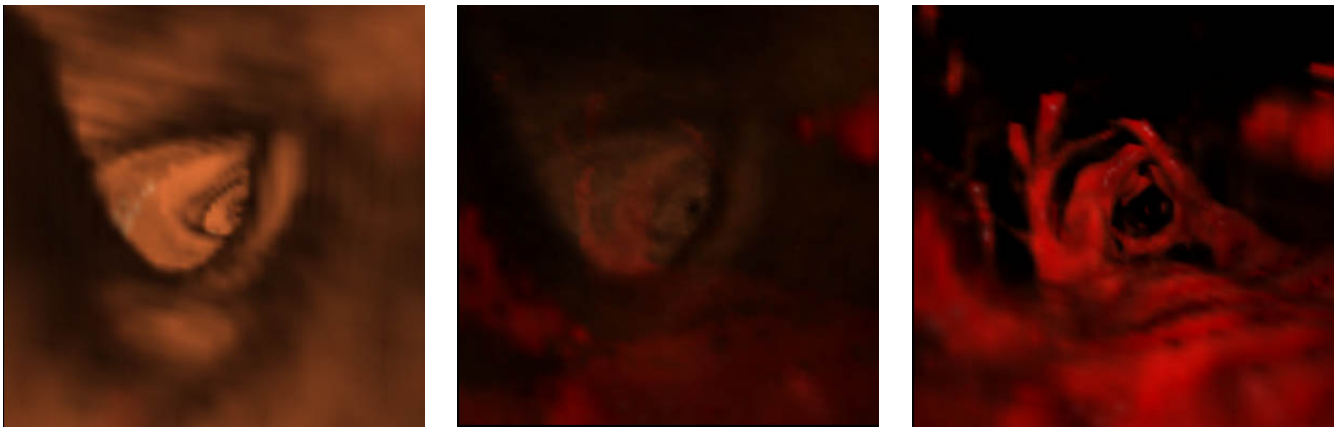
[14] M. Wan, Q. Tang, A. Kaufman, Z. Liang, and M. Wax. Volume rendering based interactive navigation within the human colon. In *IEEE Visualization '99*, pages 397–400. IEEE, nov 1999.

[15] R. Westermann and T. Ertl. Efficiently using graphics hardware in volume rendering applications. In *SIGGRAPH 98 Conference Proceedings*, pages 169–177. ACM SIGGRAPH, 1998.



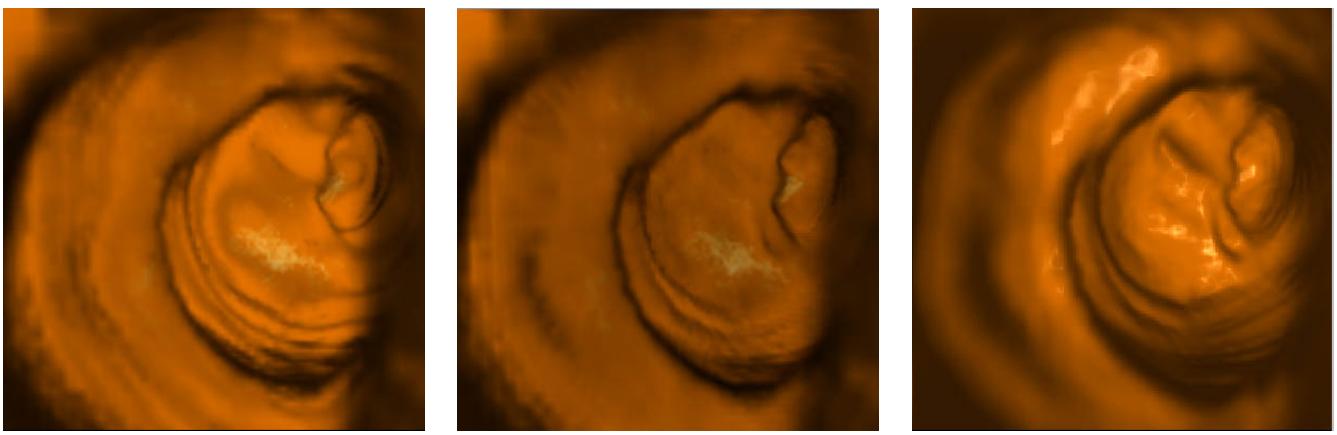
a) b) c)

Figure 6: Visualization of the CT trachea data set of a corpse compared to a real endoscopic view: **a)** projected-slabs algorithm with incremental slab thickness (37 slabs), **b)** projected-slabs algorithm with constant slab thickness equal to one voxel distance (162 slabs), **c)** real bronchoscopy snapshot.



a) b) c)

Figure 7: CT trachea data set rendered with different transfer functions. From left to right from opaque to transparent trachea walls.



a) b) c)

Figure 8: Spiral CT colon data set visualization: **a)** projected-slabs algorithm with incremental slabs thickness (28 slabs), **b)** projected-slabs algorithm with constant slab thickness and with the maximal error equal to **a** (179 slabs), **c)** brute force ray casting algorithm.