Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004)
O. Deussen, C. Hansen, D.A. Keim, D. Saupe (Editors)

Efficient Display of Background Objects for Virtual Endoscopy using Flexible First-Hit Ray Casting

A. Neubauer[1], M. Forster[2], R. Wegenkittl[3], L. Mroz[3] and K. Bühler[1]

[1] VRVis Research Center, Vienna, Austria
[2] Universitätsklinik für Neurochirurgie, Vienna, Austria
[3] Tiani Medgraph AG, Brunn/Gebirge, Austria

Abstract

Many applications of virtual endoscopy require the display of background objects behind the semi-transparent surface of the investigated organ. This paper deals with pre-processing and visualization of background objects for virtual endoscopy. A new first-hit ray casting technique for efficient perspective iso-surfacing of arbitrarily selected objects of interest is described: Visualization is performed without the use of dedicated hardware or data structures limiting flexibility (e.g., polygonal meshes or distance fields). The speedup is gained by exploiting inter-pixel coherency and by finding a near-optimal compromise between reduction of ray-tracking distances and limitation of the administrational cost associated with this reduction. The algorithm was developed by enhancing the previously published cell-based first-hit ray casting algorithm. This paper describes the original algorithm and explains the extensions needed to achieve interactive rendering of background objects.

1. Introduction

Virtual endoscopy is constantly gaining importance in modern medicine. Apart from its application as a tool for diagnosis (e.g., virtual colonoscopy), virtual endoscopy is, within the medical community, increasingly recognized as a feasible tool for pre-operative surgical planning and training. New fields of application emerge with the increasing number and importance of minimally invasive medical procedures, usually performed using endoscopes [Bar03].

An example is endonasal transsphenoidal surgery, often aimed at the removal of a tumor from the hypophysis [GRF*63]: A rigid endoscope is inserted into the patient's nose and advanced into the sinus sphenoidalis. This is a cavity inside the human skull, separated from the hypophysis only by a usually very thin bony structure, the sella floor. In order to remove the tumor, the sella floor must be opened using a bone punch. Then the tumor is cut off the surrounding tissue and removed, again through the patient's nose. This procedure is not free of danger for the patient and therefore requires the surgeon to be skilled and well-trained. An important artery (arteria carotis interna) and the optical nerve pass the hypophysis along the far side of the sella floor. They are therefore not visible to the surgeon and endangered to be damaged by the punch. The use of virtual endoscopy can help reduce this danger significantly: One of the most important advantages of virtual endoscopy is the possibility to render the interior of the investigated cavity semi-transparent, with objects of interest in the background. Using virtual endoscopy as a training or navigation device and rendering major blood vessels, the optical nerve, and the tumor behind the translucent sella floor can give the surgeon a good feeling of where to cut the sella floor (see Figure 10).

The rendering pipeline proposed by this paper is illustrated in Figure 1: An object is extracted from the volume using a segmentation technique (step 1). Then the volume is manipulated such that an iso-surface is created around the object (step 2). This iso-surface is smoothed (step 3) and finally rendered using an iso-surfacing visualization technique (step 4). This paper presents new algorithms for the pre-processing (steps 2 and 3) and efficient visualization (step 4) of background objects in virtual endoscopy applications. Section 2 discusses pre-processing of background objects. Section 3 covers related work in the field of visualization for virtual endoscopy systems. Section 4 describes a new iso-surfacing algorithm used for the visualization of background objects. Section 5 presents results. Conclusions are given in section 6.

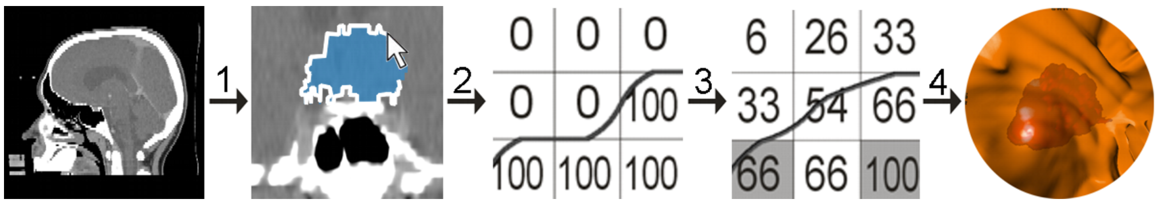

Figure 1: *The rendering pipeline. Step 1: Object segmentation; Step 2: Iso-surface generation; Step 3: Iso-surface smoothing; Step 4: Rendering*

2. Pre-Processing of Background Objects

There are two kinds of background objects in virtual endoscopy applications: The first are those which can be represented by an iso-surface in the original data volume. Examples are bones and, if a contrast agent has been used during data acquisition, also blood vessels. The second group of background objects are those which are extracted using a segmentation technique other than thresholding and therefore must be retrieved from a binary segmentation mask (e.g., the tumor). Iso-surfacing in the original data volume can, in general, not be used to render such an object. Therefore the data volume must be edited such that a smooth artificial iso-surface is constructed around the mask - the mask must be *voxelized*.

Simple voxelization techniques assign a value v_1, with $v_1 < t_{iso}$ and t_{iso} being the iso-value (the *threshold*) defining the iso-surface, to the voxels surrounding the mask and a value v_2 with $v_2 > t_{iso}$ to all voxels belonging to the mask. The resulting iso-surface, however, suffers from heavy aliasing, since the binary classification results in a discontinuous inside-outside function with an unbounded frequency spectrum. Thus a considerable fraction of the iso-surface is positioned exactly on cell boundaries, visibly revealing the underlying voxel-structure (see Figures 2 and 3). To improve image quality, the iso-surface should be faired. One way to do this is low-pass filtering [SK98]. A problem with filtering, however, is that object details are smoothed out. The straightforward solution to this is to confine voxels inside the mask to values $\geq t_{iso}$ and other voxels to values $\leq t_{iso}$. This, however, can again lead to aliasing, because, with increasing number of filtering iterations, voxel values near the object boundary tend to acquire the value t_{iso}. If values are confined to be $\geq t_{iso} + \delta$ or $\leq t_{iso} - \delta$ with an arbitrary value δ, a large number of filtering iterations yields a quasi-binary classification - voxels near the boundary acquire either the value $t_{iso} + \delta$ or the value $t_{iso} - \delta$, restoring aliasing as it was before filtering. Confining filtering to only those voxels which are very close to the object boundary reduces this effect, but still does not produce satisfying results.

It is a better idea to generate a *fuzzy boundary*, where data values near-linearly degrade or ascend from the inside of an object to the outside within a certain neighborhood of the object boundary. Lakare and Kaufman [LK03] suggest a method for transfer function-based direct volume rendering: intensities of the voxels on and near the boundary of the binary segmentation mask are altered in such a way that a fuzzy boundary is created. The following paragraph describes a similar technique. The major difference is that, since it is used for iso-surfacing, special care has to be taken that the surface retains the principal shape of the binary mask:

Each voxel near the object boundary is assigned a new value v, with $v_1 \leq v \leq v_2$ and $v_2 - t_{iso} = t_{iso} - v_1$ using the following algorithm (see Figure 2 for an example): A *reference mask* is established by eroding the segmentation mask n times. After each erosion step, those voxels that have been removed are checked, whether they are still adjacent to the eroded object. If that is not the case for a certain voxel V after the ith erosion step, V is returned to the object and not subject to erosion any more. It is therefore part of the reference mask. Its value v is set to

$$v = v_2 - (v_2 - v_1) \cdot \frac{n+1-i}{2n+1} \tag{1}$$

The voxel thus acquires a value larger than t_{iso} (because V is part of the object) and smaller than v_2 (because V has fallen victim to erosion). The nearer V is to the object boundary, the smaller is its new intensity value v. All voxels inside the final reference mask, which have not fallen victim to erosion, are assigned the value v_2. Then $2n$ dilation operations are performed. The boundary voxels of the reference mask are tracked as *reference voxels* with the dilation front. Each voxel V that is added through dilation is therefore associated with a reference voxel V_{ref} and acquires the value

$$v = v_{ref} - (v_2 - v_1) \cdot \frac{d}{2n+1} \tag{2}$$

with d being the Euclidean distance from V to V_{ref} and v_{ref} the value of voxel V_{ref}. If voxel V receives two or more reference voxels (this happens due to concave parts of the object), the one which yields the largest voxel value for V is chosen. This algorithm yields a near-linear value degradation from the reference mask outwards and therefore a considerably smoothed iso-surface (see Figure 2). Each point of the iso-surface is located at a distance of $(n+1)/2$ from the nearest voxel V_{ref} of the reference mask, if V_{ref} has the

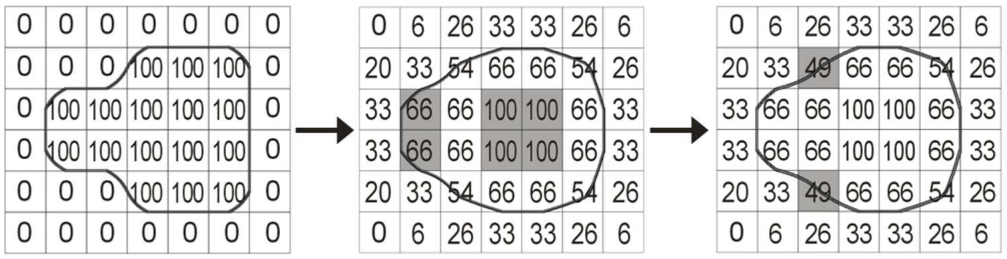

Figure 2: *Surface smoothing:* $v_1 = 0$, $v_2 = 100$, $t_{iso} = 50$. *Left: Iso-surface initialized; Center: Iso-surface smoothed (highlighted voxels are part of the reference mask - those which remained after the erosion were set to the value* 100*, the others acquired their values through equation 1); Right: Erroneously added voxels (highlighted) are removed from the object*

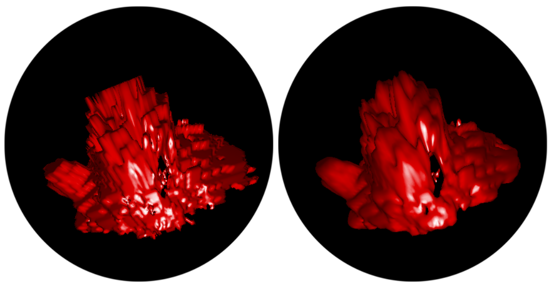

Figure 3: *Left: Object imported from binary mask without smoothing; Right: Smoothed object*

value v_2. Otherwise (if V_{ref} has acquired its value through equation 1), the distance is proportionally smaller. The principal shape of the object is therefore maintained. Only single voxels are removed from or added to the object - their number can be bounded by selecting a low value n. Also they can be corrected by setting their values to slightly larger or smaller than t_{iso}. Figure 2 shows an example: The original iso-surface is smoothed ($n = 1$) and voxel values are corrected. For the results given in this paper, $n = 1$ was used.

3. Visualization: Related Work

To be applicable, a virtual endoscopy system must allow for visualization of both foreground and background at interactive frame rates. Concise overviews of visualization techniques used for virtual endoscopy are given by Vilanova [Vil01] and Bartz [Bar03]. For reasons of rendering complexity, most systems depict iso-surfaces instead of using transfer-function-based direct volume rendering. Roughly two groups of iso-surfacing algorithms can be identified: those which use hardware-accelerated polygonal surface rendering and those using software-based first-hit ray casting:

Polygonal surfaces are often extracted using the marching cubes algorithm [LC87] which, however, often yields a polygon mesh too complex to be rendered at real time. This problem is overcome by the use of geometry simplification [Hop96] and occlusion culling [HMK*97, HO00]. Another drawback, however, remains: Due to the costly surface extraction process, it is not possible to adjust the iso-value of the surface at interactive frame rates. This can be a problem in many applications, since interactive threshold adjustment generally improves the information output.

An alternative to using pre-computed polygonal surfaces is to perform iso-surfacing using perspective first-hit ray casting [Lev88]. Due to the heavy CPU load induced by perspective software ray casting, finding an applicable technique has always been a task of finding a compromise between rendering speed and flexibility or image quality. Distance fields [ZKV92, FS97] are a well-known method of accelerating ray-traversal through empty spaces: Each voxel is assigned its distance to the nearest point of the iso-surface - this distance can be skipped by a ray passing the voxel. The distance field, however, must be reprocessed whenever the threshold is changed, which heavily reduces the frame rate during threshold adjustment. Also, distance fields require a considerable amount of memory. A more flexible approach is the so-called Lipschitz method [SH95]: The data value at the current ray position, the maximum data gradient inside the currently traversed sub-volume and the threshold are used to calculate a minimum distance to the iso-surface on-the-fly. Changes of the threshold therefore do not induce any performance loss. The distances that can be skipped, however, tend to be rather small, limiting the gain of computation speed. Also there are a number of approaches which pay for increased performance by reducing image quality: Vilanova et al. [VWKG01] and Kreeger et al. [KBD*98] introduced a technique which simulates perspective whilst tracing only parallel ray-segments. A completely different approach of accelerating perspective ray casting is image-based rendering. Qu et al. introduced a technique they called *keyframe-less rendering* [QWQK00]: For each frame, the image of the previous frame is warped such that it best fits the current viewing parameters. Ray casting is only used to fill gaps in the image. Speedup is gained at the expense of visual artefacts. Wegenkittl et al. [WVH*00] proposed a method which achieves highly interactive frame rates, but confines the camera position to a predefined path: Virtual cubes are centered on discrete points along the path. In a pre-processing phase, an image, containing also background objects of interest, is rendered to each of the six sides of each virtual cube, using direct volume rendering. During the interaction phase, it is possible to rotate the camera arbitrarily by accordingly projecting the six sides of the cube associated with the current view point to the screen. The visual appearance of investigated structures, however, cannot be adjusted during interaction.

Foreground and background can be rendered using different visualization techniques. A combination of ray casting for the foreground and polygon rendering for background objects is imaginable. Still, it is often desired to have the possibility of interactive threshold adjustment also for each background object - especially when background objects are rendered, which do not stem from binary segmentation masks, e.g., bone structures or contrasted blood vessels. This reduces the need for tedious segmentation - segmentation can be carried out during the 3D-investigation using interactive threshold adjustment. Another advantage of first-hit ray casting over polygon rendering is that complex objects are always rendered at the highest possible level of detail (LOD), while the LOD of a polygon mesh representing a certain object often has to be adapted [Hop96] during the virtual investigation in order not to waste rendering time during phases when the object is still far away from the view point (see color plate, Figure 10). This again requires a lot of preprocessing and increases memory load.

For creating some of the images in this paper, a combination of two different ray casting techniques was used: Object-order ray casting was used for displaying background objects and image-order ray casting was used for foreground rendering. The images acquired by the two rendering passes were combined using a Z-Buffer. The following section describes the new algorithm for background rendering.

4. Visualization: The New Algorithm

In this section, the generation of the background image will be explained. The basic idea of the algorithm used is taken from the previously published *cell-based first-hit ray casting* technique [NMHW02]. This algorithm works in an object-order fashion and is therefore well suited especially for the display of background objects. Section 4.1 will briefly describe cell-based first-hit ray casting. Sections 4.2 through 4.6 will, in more detail, explain the changes and optimizations that were applied.

4.1. Cell-based first-hit ray casting

The volume extracted from the CT scan of the patient's head is divided into cubic bricks, referred to as *macro-cells*. These bricks form the leaves of a min-max-octree, implemented as an integer-array as proposed by Wilhelms and van Gelder [WG90]. This octree can be efficiently traversed to find all macro-cells containing a part of an iso-surface in order of increasing distance from the eye-point. Each of those macro-cells is then projected to the image plane. The projection is rasterized to the screen, scan line by scan line. For each pixel which is covered by the projection and has not yet been assigned a color, a local ray segment is tracked inside the macro-cell using the fast voxel traversal technique by Amanatides and Woo [AW87]: The ray position is efficiently propagated from one cell-boundary to the next, therefore quickly identifying all cells pierced by the ray. This traversal algorithm requires costly calculations solely for ray initializations. When a cell which contains a part of an iso-surface is encountered, an intersection test is performed. If an intersection exists, the surface normal at the intersection point is calculated and the pixel is shaded accordingly. Some optimizations of the algorithm were previously proposed [NMHW02]:

Macro-cell trimming Instead of the complete macro-cell, only the smallest cuboid containing the iso-surface inside the macro-cell is projected, decreasing the number of local ray segments having to be tracked. Also, local ray segments are confined to this smallest cuboid. This latter optimization was partly discarded to allow for a more effective one described in section 4.3.

Early scan line termination Early scan line termination is a heuristic measure yielding a reduction of the number of ray segments tracked. Based on an analysis of the iso-surface geometry inside the macro-cell, a decision is made, whether rasterization of a scan line can be stopped as soon as a ray segment did not intersect the iso-surface. Due to the heuristic character, errors can occur. It was shown how these errors can be efficiently detected and corrected [NMHW02]. Unfortunately, early scan line termination is in general less effective when visualizing only background objects, compared to its application for generating the foreground for a virtual endoscopy system (see also section 4.6).

Screen regions Small rectangular regions of the screen which have already been completely filled, can be skipped by rasterization.

The main principle of acceleration in cell-based first-hit ray casting is the reduction of ray traversal lengths. Sections 4.3 and 4.6 will introduce optimization techniques which increase the number of ray steps tracked, but still speed up rendering by finding compromises satisfying the trade-off between the ray traversal time saved and the cost associated with saving it. Sections 4.2, 4.4 and 4.5 will explain, how pixel-space coherency can be exploited for further speed-up. Figure 4 shows an overview of which steps of the original method are adapted to allow for the proposed optimizations.

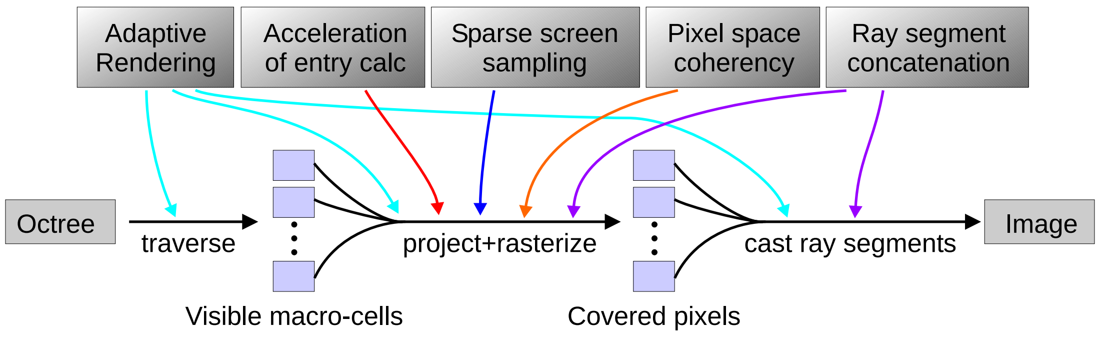

Figure 4: *The new optimizations and which parts of the original algorithm they require to be changed*

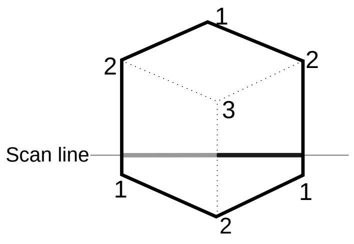

Figure 5: *Silhouette lines (solid), frontier lines (dotted), visibility coefficients and an example scan line*

4.2. Acceleration of entry point calculation

The starting point of a local ray segment (the entry point) is found by intersecting the ray with the macro-cell. If it is known, through which face (on which side) the ray enters the macro-cell, only a ray-plane (instead of ray-cuboid) intersection must be calculated. Using a simple algorithm which determines the entry face during rasterization can therefore save valuable rendering time: After a macro-cell has been projected to the image plane, a rasterization process, traversing the projection scan line by scan line, determines the set of pixels, for which local ray segments have to be tracked through the macro-cell. Depending on the current view point, three faces of the current macro-cell are visible at most. Each projected macro-cell vertex is assigned a *visibility coefficient* indicating how many of the faces adjacent to the vertex are visible. If the visibility coefficient is 1 or 2, the vertex is part of the outline of the projection. Those vertices are connected by *silhouette lines*. Vertices with visibility coefficients of 2 or 3 define the end points of *frontier lines*. Figure 5 shows an example projection, illustrating silhouette lines, frontier lines and visibility coefficients. Each scan line is first intersected with all silhouette lines to determine the set of pixels that have to be processed. Then the scan line is intersected with all frontier lines to determine the entry face for each local ray segment (see Figure 5).

The overall rendering time saved due to the faster intersection tests amounts to about 8 percent on the average.

More significant acceleration might be achieved by calculating entry points using texture-mapping hardware, using, for example, the technique proposed by Westermann and Sevenich [WS01]. This, however, has yet to be tested.

4.3. Ray segment concatenation

Often a ray moves along the surface of an object for a long distance, and through many macro-cells, until it finally intersects the iso-surface, if it does that at all. The actually tracked portion of such a ray consists of a considerable amount of local ray segments, each having to be initialized. Ray segment initialization, including for example entry point calculation and the calculation of parameters used for the fast voxel traversal, is, summed over all ray segments during one frame, a costly part of the algorithm. Therefore it is more efficient to concatenate consecutive ray segments, wherever possible: Whenever a ray leaves the macro-cell, it is checked, whether the macro-cell which is now entered contains a part of the iso-surface. In the positive case, the ray segment is tracked further, otherwise ray tracking is stopped. This algorithm requires some simple extensions of the data structure: So far, the information whether a macro-cell contains a part of an iso-surface or not is only coded in the min-max values at the leaves of the octree, where neighborhood information cannot be easily retrieved. Therefore an additional bit-volume is needed, with one bit per macro-cell, indicating, whether the macro-cell contains a part of an iso-surface. If the bit is set, the macro-cell will be entered by local ray segments arriving at its boundary. Another extension to the data structure is needed to avoid ray segments being traced more than once: For each pixel, a value *concat* is stored. It indicates the number of macro-cells having been traversed in advance for this pixel. Whenever a ray segment is stopped without having intersected the iso-surface, the value *concat* of the according pixel is set. The new value is calculated by checking for each macro-cell that the ray segment traverses, except the one in which the ray segment

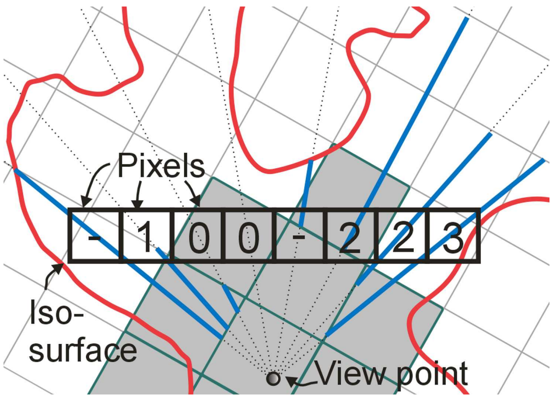

Figure 6: *Ray segment concatenation and* concat *values*

was started, whether the portion of the macro-cell, that will be projected (after macro-cell trimming), is pierced by the ray, and counting the macro-cells for which this is the case.

Figure 6 illustrates ray segment concatenation and the calculation of the *concat* values with a 2D-example. Here it is assumed that macro-cell trimming has not been performed and only whole macro-cells are projected. Squares filled with gray color correspond to macro-cells which have already been processed. The viewing plane consists of eight pixels, the numbers indicate the current *concat* value of each pixel ("-" means the pixel has acquired its color already). A viewing ray (dotted line) connects the eye point to the center of each pixel. The bold portions of the viewing rays are the parts which have actually been tracked up to this point in time.

Ray segment concatenation causes the rasterization algorithm to change slightly: For each pixel that is still empty and inside the projection of the currently processed macro-cell, first the *concat* value is checked. Only if it is zero, a local ray segment is started, otherwise *concat* is decreased by 1.

A drawback of this approach is that now ray segments must be tracked all the way to the boundary of the current macro-cell in order to find the entry point to the neighboring macro-cell. Ray tracking can no more be confined to the portion of the macro-cell remaining after trimming. Also, it has to be noted, that not all ray segments that are concatenated to a long ray segment would, if ray segment concatenation was not applied, be tracked. Some would be dropped due to macro-cell trimming or early scan line termination. Still, the impact of this problem is clearly outweighed by the performance gain brought about by ray segment concatenation: The number of ray segments is reduced by about 59 percent. Despite an increase of tracked ray steps by about 6 percent, rendering time decreases by about 17 percent on average.

Figure 7 shows the effect of ray segment concatenation on ray tracking distance, with macro-cell trimming having

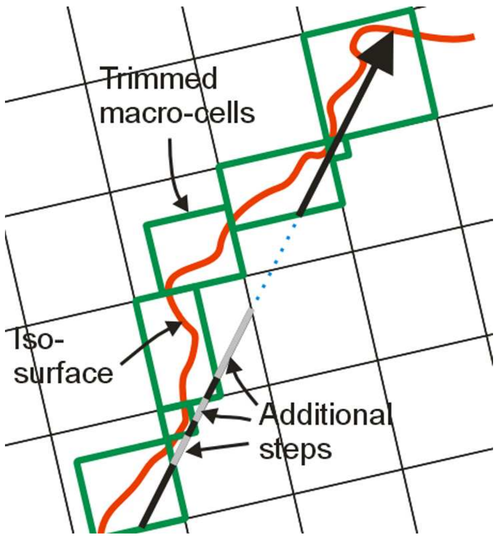

Figure 7: *Increased ray-tracking distance caused by ray segment concatenation*

been performed. Bold parts of the depicted ray are actually tracked. In this example, ray segment concatenation reduces the number of tracked ray segments from 6 to 2. The downside is the increased number of tracked ray steps: The parts of the ray which are coloured gray would not be tracked in the original algorithm.

4.4. Sparse screen sampling

The number of local ray segments which fail to intersect the iso-surface has already been significantly reduced by applying ray-segment concatenation, macro-cell trimming and early scan line termination. Still, there remains a considerable number of ray segments cast in vain, especially when the eye point is very close to a visualized background object. There are two main reasons for this: Background objects can be arbitrarily small and arbitrarily rough, therefore surface normals may vary at high frequencies, resulting in a large variety of gradient directions inside a macro-cell. This can prohibit early scan line termination completely or confine it to only a small number of scan lines. The second main reason is the considerable magnification induced by perspective rendering, resulting in empty spaces inside a macro-cell being projected to a large portion of the screen - the projection of a macro-cell can in fact cover the complete screen, if the eye point is inside or very close to the macro-cell. Traversing big empty regions pixel by pixel is computationally expensive. Applying a sparse sampling strategy can significantly reduce this problem. Reduction of sampling frequency, however, is always associated with some amount of quality reduction, which should be bounded, keeping the following things in mind:

- The quality of the visual output is heavily affected, if pixels defining parts of the outline of a visualized object are not computed correctly.
- The information loss associated with an error in the image is directly proportional to the relation of the size of the erroneous region to the sizes of the projections of structures holding the main image information. A small erroneous region could, for example, completely hide an object whose projection is very small - but it would do only little harm to the information given by a big projection of an object.

In the algorithm described by this paper, sampling frequency is determined by two parameters: $skip_{column}$, the number of pixels skipped by the rasterization process after a local ray segment has missed the iso-surface, and $skip_{row}$, the number of scan lines skipped, after all local ray segments of a scan line have missed the iso-surface (see pseudo code in Figure 9). As soon as an intersection has been found and therefore a pixel has been assigned its color, the pixels that were skipped between the previous and the current ray segments have to be processed to find the real object boundary. Also, for the same reason, scan lines skipped between the previously processed scan line and the current one must be processed, if during rasterization of the current scan line intersections with the iso-surface have been found. This ensures that the boundaries of all displayed objects are depicted correctly. Due to ray segment concatenation, the probability of parts of objects being lost due to skipping pixels is initially very low. It can be further reduced by tracking ray segments backwards, as illustrated in Figure 8. Complete objects whose projections never intersect the sparse sampling grid, might, however, be lost. In order to keep information loss at a minimum, values $skip_{column}$ and $skip_{row}$ can be adapted to the size of the projection of the macro-cell (see also section 4.6). If a macro-cell is projected to a large portion of the screen, one can expect that important structures inside this macro-cell will appear large in the final image. Therefore a coarser sampling grid can be applied. For macro-cells which are far from the view point, the skip values should be kept low in order to find all important objects. This restriction is bearable: As pointed out above, sparse sampling is effective especially for macro-cells close to the eye point - larger skip values for far macro-cells would not significantly reduce rendering times. For the results given in this paper, $skip_{column}$ and $skip_{row}$ were set to 4 and 1, respectively, for near macro-cells and to 1 and 1, respectively, for far macro-cells. The probability of important objects being lost is therefore extremely low. Sparse screen sampling saves another 23 percent of rendering time on average.

4.5. Exploiting pixel space coherency

The number of local ray segments can further be reduced by exploiting the well-known heuristic that a pixel whose neighbors all have similar color can be expected to have a color similar to its neighbors. The following strategy is applied: Whenever a local ray segment has hit the iso-surface and the according pixel is assigned its final intensity i_1, rasterization skips one pixel and the next local ray segment is cast for the next but one pixel in the currently processed scan line. If the resulting intensity i_2 fulfills the similarity criterion

$$| i_1 - i_2 | < t_{sim}$$

with the predefined threshold t_{sim}, the pixel that was skipped acquires its color through linear interpolation. Otherwise, a local ray segment is cast also for this pixel. After a scan line has been finished, one scan line is skipped and the next but one scan line is processed. After that, rasterization returns to the scan line that was skipped. As pointed out in the previous section, it is important to correctly compute pixels depicting the outline of the object. Therefore, now ray segments are cast for each pixels that is still empty and has exactly one vertical neighbor that has been assigned its final color. If both vertical neighbors are filled, the similarity criterion is used to decide whether a ray segment is cast, or linear interpolation is applied. This strategy saves an average of 8 percent of overall rendering time. A pseudo-code implementation of the rasterization process including sparse sampling and exploitation of pixel-space coherence is given in Figure 9.

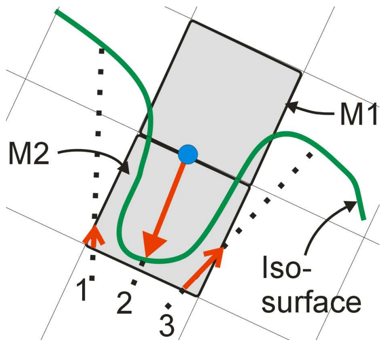

Figure 8: *Tracking ray segments backwards: During processing of macro-cell M2, rays 1 and 3 encounter no intersection, therefore ray 2 is erroneously skipped. Macro-cell M1 is processed after M2. The entry point of ray 2 into M1 is inside the object - the ray must be traced backwards.*

4.6. Adaptive rendering

The algorithm described so far computes all macro-cells in the same way. Only the sampling frequency parameters introduced in section 4.4 differ between macro-cells, depending on the sizes of the macro-cell projections. This section shows some more examples of how the size s_p of a macro-cell projection can be used to increase flexibility of the algorithm in order to maximize performance:

```
For all scan lines from top to bottom:
    For each pixel from left to right projection boundary:
        Cast ray segment, if the pixel is empty
        Pixel still empty ?
            No:
                How many pixels were skipped before?
                    1:  Are the neighbors of this pixel similar?
                            yes: Interpolate pixel color
                            no: Cast ray segment
                    >1:
                        For each skipped pixel:
                            Cast ray segment
                Skip 1 pixel
            Yes:
                Skip $skip_{column}$ pixels
    Has at least 1 intersection been found ?
        Yes:
            How many scan lines were skipped before?
                1: For each pixel in skipped scan line:
                        How many vertical neighbors filled ?
                            0:  Do nothing
                            1:  Cast ray segment
                            2:  Are the two neighbors similar ?
                                    yes: Interpolate pixel color
                                    no: Cast ray segment
                >1: Rasterize skipped scan lines
        No: Skip $skip_{row}$ scan lines
```

Figure 9: *Pseudo code of the rasterization process*

Early scan line termination The performance gain brought about by early scan line termination is not for free: Some computation has to be performed for each macro-cell in order to identify the set of scan lines for which early termination is applicable. This includes calculating a gradient vector for each surface cell inside the macro-cell and projecting it to screen space. The reward is a reduced number of ray segments having to be tracked. A reduction of per-pixel-overhead is achieved at the expense of increased per-macro-cell-overhead. If the number of ray segments can be reduced only slightly or not at all (because it is found that early scan line termination is not applicable), processing of the macro-cell becomes more expensive, instead of cheaper. This is more likely to happen if a macro-cell is projected to only a small region of the screen, resulting in only few ray segments in the first place. Therefore it is more efficient to confine the application of early scan line termination to macro-cells near the eye-point. Since, due to the octree traversal scheme, macro-cells are processed in order of increasing distance from the eye point, the application of early scan line termination can simply be turned off at some point during the traversal, optimally as soon as the parameter s_p of a processed macro-cell is below a predefined threshold t_{est}.

Macro-cell size Both large and small macro-cells have their advantages and disadvantages: A small macro-cell size leads to a large number of macro-cells, which increases the time needed to traverse the octree and perform computations required for each macro-cell, e.g., rasterization. On the other hand, small macro-cells offer a more compact and more accurate representation of the visualized object and, despite the balancing effect of macro-cell trimming, generally yield fewer local ray segments, especially for smaller objects. To exploit the advantages of both small and large macro-cells, small macro-cells are used near the eye-point, where the small macro-cell size can effectively reduce the number of ray segments, while in regions far from the eye point the usage of larger macro-cells is more efficient. An arbitrary change of macro-cell size would result in the need to completely rebuild the min-max octree, which is not possible at interactive frame rates. Flexibility in this matter is therefore restricted to treating the sub-volumes represented by nodes at differing levels of the octree as macro-cells. The following strategy is used: From the moment when a macro-cell with $s_p < t_{macro}$ for a predefined threshold t_{macro} has been encountered, sub-volumes represented by nodes of the second lowest level (instead of the lowest level) of the octree are used as macro-cells, increasing the macro-cell size by factor 2 in each dimension. Using $t_{macro} = t_{est}$ makes sense, because early scan line termination is more effective for small macro-cells. For the results reported in this paper, macro-cell size was $4 \times 4 \times 4$ voxels initially and extended to $8 \times 8 \times 8$ voxels for far objects.

Sparse space sampling The fast voxel traversal algorithm applied for tracking a ray segment detects each single cell inside the macro-cell that is intersected by the ray to check for an intersection with the iso-surface. This level of accuracy is important for parts of the iso-surface that are close to the view point. Otherwise, disturbing visual artefacts would be likely to appear. For distant objects, sampling frequency along the ray can be reduced significantly without creating disturbing image errors. Thus, from the moment when s_p is smaller than a predefined threshold t_{samp} for the first time, the resolution inside the macro-cell is coarsened by combining eight cells to one. This can be done efficiently by manipulating the parameters of the fast voxel traversal algorithm.

Applying these adaptive rendering strategies saves about 20 percent of rendering time on average. The parameter s_p should be an efficiently computable estimate of the size of the projection of the macro-cell. Since the projection size is indirectly proportional to the distance of the center of the macro-cell, simply calculating the (squared) distance would be a feasible measure. To allow for interactive adjustment of the opening angle of the viewing frustum without loss of performance, s_p should be normalized with respect to the opening angle. For the results presented in this paper, a different method is used: The (quadratic) length of the diagonal of the smallest bounding box of the projection is calculated and normalized with respect to the (quadratic) distances between each pair of vertices whose projections define the side

	P4 1900		
	min	max	av.
Tumor	6.1	19.4	11.0
Tumor and Vessels	4.5	15.4	8.6
	P4 3000		
Tumor	11.1	32.3	18.6
Tumor and Vessels	8.0	25.1	15.1

Table 1: *Minimum, maximum and average frame rates (in frames/second) during a virtual endonasal examination*

lengths of the bounding box. This normalization has to be done for two reasons: The first reason is that due to macro-cell trimming (and the flexible macro-cell sizes introduced in this section) sub-volumes of various sizes are projected. The second reason is that this normalization helps to compensate for the fact that the rotational position of the macro-cell relatively to the viewing frustum also has significant impact on the size of the bounding box.

5. Results

The algorithm described in this paper was implemented in Java and tested on two different systems: an Intel P4 1900 and an Intel P4 3000 single processor machine. All timings given in this section were measured using a $512 \times 512 \times 201$ data set of a human head. Table 5 gives timings acquired during a complete virtual endonasal investigation (starting inside the nose and ending inside the sinus sphenoidalis), giving minimum, maximum and average performance of background visualization for a circular image with a diameter of 429 pixels. For the investigation two background objects were loaded from binary masks: the tumor and a set of important blood vessels, respectively. The examination started inside the nose - with the background objects still far away, naturally, the highest frame rates were achieved - and ended just in front of the sella floor (see color plate, Figure 10), where background objects cover a large fraction of the screen (the maximum was 78 percent) and the lowest frame rates were measured.

Table 2 shows the impact of each optimization presented in this paper on the average performance and average numbers of ray segments and ray steps when rendering two background objects.

Due to the object-order nature of the algorithm, performance is highly dependent on the sizes of the object projections. The performance drops with increasing number of objects and with decreasing distance of the objects to the view point due to the bigger amount of pixels having to be processed.

6. Conclusion

This paper dealt with efficient and flexible visualization of background objects for virtual endoscopy. Pre-processing and visualization of background objects were covered. Visualization is performed using first-hit ray casting. The algorithm renders background objects for virtual endoscopy applications at interactive frame rates on standard single-CPU PCs without the use of hardware-acceleration and without limitations of flexibility. The technique is based on cell-based first-hit ray casting, an algorithm which gains computational speedup mainly through effective reduction of ray steps through empty parts of the volume. This speed-up is achieved at the expense of increased administrational overhead. With some of the improvements introduced in this paper, this administrational overhead can be significantly reduced, whilst still keeping ray step numbers at a low level. Additionally, inter-pixel coherence can be effectively exploited to considerably reduce the number of ray steps.

The algorithm was tested and assessed within the scope of a medical dissertation [For03], where it was recognized as being feasible for clinical practice. Further development of the system in close cooperation with medical partners is planned for the near future.

Acknowledgements

This work has been carried out as part of the research project 'Virtual Reality for Scientific Applications' at the VRVis research center (http://www.vrvis.at) in Vienna, Austria, which is funded by the Austrian governmental research project Kplus. Thanks to Tiani Medgraph AG (http://www.tiani.com) for providing useful implementation facilities.

References

[AW87] AMANATIDES J., WOO A.: A fast voxel traversal algorithm for ray tracing. In *Proc. of Eurographics '87* (1987), pp. 3–10. 4

[Bar03] BARTZ D.: Virtual endoscopy in research and climical practice. In *Eurographics State-of-the-Art-Reports 2003, (S4)* (2003). 1, 3

[For03] FORSTER M.-T.: *Virtuelle Endoskopie in der transnasalen Transsphenoidalen Hypophysenchirurgie*. PhD thesis, Universität Wien, 2003. 9

[FS97] FREUND J., SLOAN K.: Accelerated volume rendering using homogeneous region encoding. In *Proc. of IEEE Visualization '97* (1997), pp. 191–196. 3

[GRF*63] GUIOT G., ROUGERIE J., FOURESTLER A., FOURNIER A., COMOY C., VULMIERE J.,

Algorithm	no. ray segments	no. ray steps	fps (P4 1900)	fps (P4 3000)
cell-based first-hit ray casting	166,875	834,622	3.7	6.7
+ faster entry point calculation	166,875	834,622	4.0	7.3
+ ray segment concatenation	68,359	887,283	4.8	8.7
+ sparse screen sampling	46,268	597,847	6.2	11.2
+ pixel space coherence	30,174	390,894	6.8	12.2
+ adaptive rendering	36,911	475,537	8.6	15.1

Table 2: *Impact of different optimizations on the average rendering performance, the number of local ray segments and the number of ray steps*

GROUX R.: Une nouvelle technique endoscopique: Exploration endoscopiques intracraniennes. *La Presse Medicale*, 71 (1963), 1225–1228. 1

[HMK*97] HONG L., MURAKI S., KAUFMAN A., BARTZ D., HE T.: Virtual voyage: interactive navigation in the human colon. In *Proc. of International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 97)* (1997), pp. 27–34. 3

[HO00] HIETALA R., OIKARINEN J.: A visibility determination algorithm for interactive virtual endoscopy. In *Proc. of IEEE Visualization ' 00* (2000), pp. 29–36. 3

[Hop96] HOPPE H.: Progressive meshes. *Computer Graphics 30*, Annual Conference Series (1996), 99–108. 3, 4

[KBD*98] KREEGER K., BITTER I., DACHILLE F., CHEN B., KAUFMAN A.: Adaptive perspective ray casting. In *Proc. of Symposium on Volume Visualization* (1998), pp. 55–62. 3

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: a high resolution 3d surface construction algorithm. In *Proc. of SIGGRAPH'87* (1987), pp. 163–169. 3

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 5 (1988), 29–37. 3

[LK03] LAKARE S., KAUFMAN A.: Anti-aliased volume extraction. In *Proc. Eurographics/IEEE TCVG Symposium on Visualization (VisSym '03)* (2003). 2

[NMHW02] NEUBAUER A., MROZ L., HAUSER H., WEGENKITTL R.: Cell-based first-hit ray casting. In *Proc. Eurographics/IEEE TCVG Symposium on Visualization (VisSym '02)* (2002). 4

[QWQK00] QU H., WAN M., QIN J., KAUFMAN A.: Image based rendering with stable frame rates. In *Proc. of IEEE Visualization 2000* (2000), pp. 251–258. 3

[SH95] STANDER B. T., HART J. C.: A Lipschitz method for accelerated volume rendering. In *Proc. of IEEE Visualization '95* (1995), pp. 107–114. 3

[SK98] SRAMEK M., KAUFMAN A.: Object voxelization by filtering. In *Proc. of IEEE Symposium on Volume Visualization* (1998), pp. 111–118. 2

[Vil01] VILANOVA BARTROLI A.: *Visualization Techniques for Virtual Endoscopy*. PhD thesis, Vienna University of Technology, 2001. 3

[VWKG01] VILANOVA BARTROLI A., WEGENKITTL R., KÖNIG A., GRÖLLER E.: Perspective projection through parallely projected slabs for virtual endoscopy. In *Proc. of SCCG '01-Spring Conference on Computer Graphics* (April 2001), pp. 287–295. 3

[WG90] WILHELMS J., GELDER A. V.: Octrees for faster isosurface generation (extended abstract). *Computer Graphics 24*, 5 (November 1990), 57–62. 4

[WS01] WESTERMANN R., SEVENICH B.: Accelerated volume ray-casting using texture mapping. In *Proc. of IEEE Visualization 2001* (2001), pp. 271–278. 5

[WVH*00] WEGENKITTL R., VILANOVA A., HEGEDÜS B., WAGNER D., FREUND M. C., GRÖLLER M. E.: Mastering interactive virtual bronchioscopy on a low-end PC. In *Proc. of IEEE Visualization 2000* (2000), pp. 461–464. 3

[ZKV92] ZUIDERVELD K. J., KONING A. H. J., VIERGEVER M. A.: Acceleration of ray-casting using 3d distance transforms. In *Visualization in Biomedical Computing II, Proc. SPIE 1808* (1992), pp. 324–335. 3

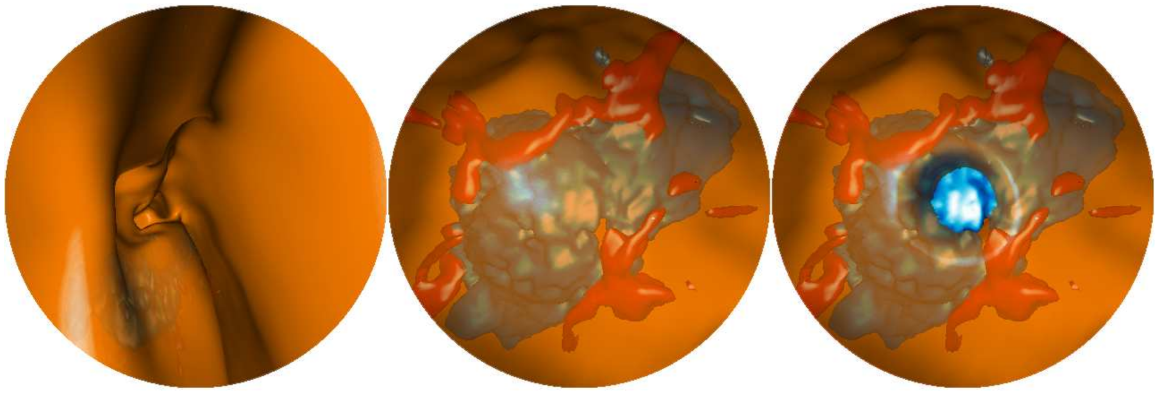

Figure 10: *Left: Moving through the nose, the tumor still far away; Center: Inside the sinus sphenoidalis; Right: The sella floor has been opened*

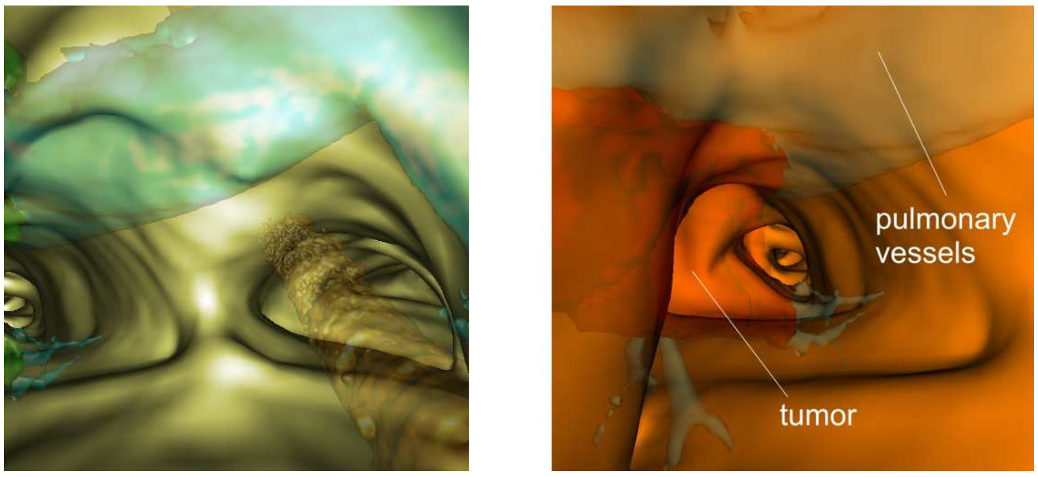

Figure 11: *Virtual bronchoscopy: Background objects are a tumor, the pulmonary artery and the aorta*

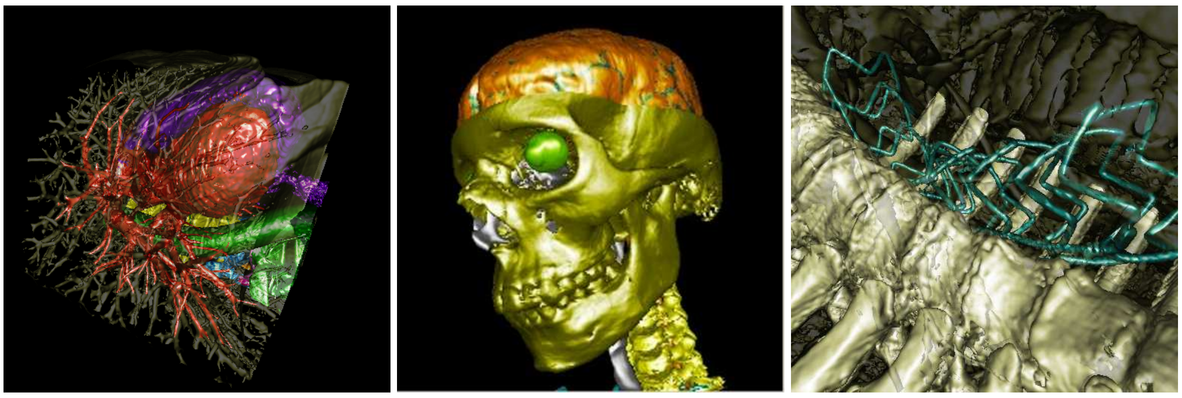

Figure 12: *Other applications: Left: A segmented heart; Center: A segmented head; Right: Assessment of stent placement*