XYZ Event (2003), pp. 1–11
A. Chairpersona, B. Chairpersonb (Editors)

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

A. Neubauer† und ein paar andere Menschen

Abstract

Many applications of virtual endoscopy require the display of background objects behind the semi-transparent surface of the investigated organ. This paper describes a new first-hit ray casting technique for efficient perspective iso-surfacing of arbitrarily selected objects of interest. 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 finding a near-optimal compromise between reduction of ray-tracking distances and limitation of the administrational cost associated with this reduction, e.g., traversal of special data structures. 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. Also, pre-processing of background objects defined by binary segmentation masks is discussed.

1. Introduction

Virtual endoscopy is constantly gaining importance in modern medicine. Apart from its application as a tool for diagnosis (eg. 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 transphenoidal 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 pushed into the sinus sphenoidalis, a cavity inside the human skull, separated from the hypophysis only by the sella floor, a bony structure which is usually very thin. 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 trained well. Important arteries (eg., arteria carotis, arteria cerebri) and the optical nerve pass the hypophysis along the far side of the sella floor and 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-transparently, with objects of interest in the background. Using virtual endoscopy as a training or navigation device and rendering the 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.

To be applicable, a virtual endoscopy system must allow for visualization of both foreground and background at interactive frame rates. Therefore a fast visualization technique with bounded reduction of rendering quality is required. Concise overviews of visualization techniques used for virtual endoscopy are given by Vilanova [Bar01] and Bartz [Bar03]. For reasons of rendering complexity, most systems depict iso-surfaces instead of using transfer-function-based direct volume rendering. An iso-surface is the set of all points (x,y,z) in the data volume which are assigned the value T (also called the *threshold*) by the function $\rho(x,y,z)$. The most commonly used fuction $\rho(x,y,z)$ is trilinear interpolation between the original voxel values. 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:

† VRVis Research Center, Vienna, Austria

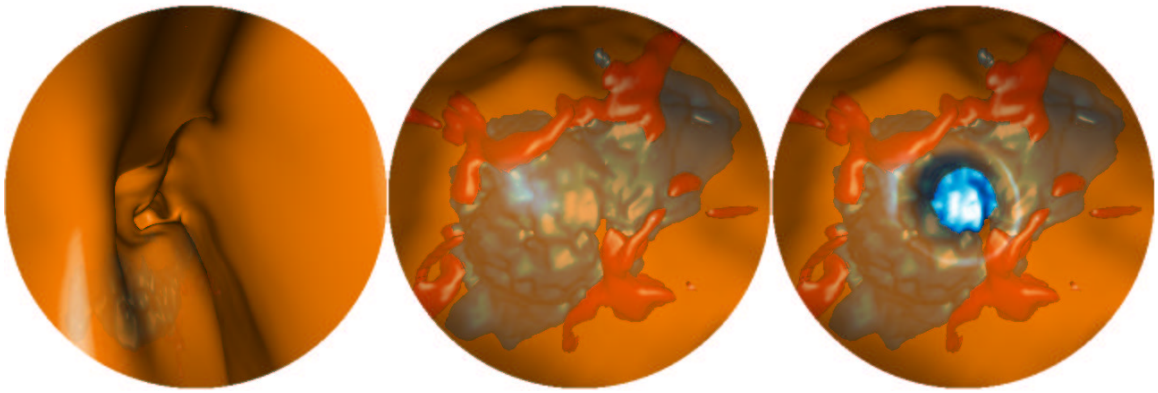

Figure 1: *left: moving through the nose, the tumor still far away; center: inside the sinus spenoidalis; right: the sella floor has been opened*

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 [BS99, 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, for example the one explained above: Inside the cavities of the skull, a small change of the threshold can affect the visual outcome tremendously. Therefore a standard threshold value or one selected by examining the plain CT image might easily prove unusable. Also, interactive threshold adjustment generally improves the information output. Hietala et al. presented a more flexible algorithm which uses sparse ray casting to find visible cells, triangulates cells that have become visible and removes those cells which ceased to be visible from the mesh. Threshold adjustment, however, still leads to heavy reduction of the frame rate, because each visible cell has to be retriangulated [HO00].

An alternative to using precomputed 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 [YS93, FS97, Sra96] 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. 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. [BWKG01] and Kreeger et al. [KBD*98] introduced a technique which simulates perspective whilst tracing only parallel ray-segments. The space in front of the virtual camera is divided into so-called *slabs*. The entry points of a ray into the slabs diverge in a perspective fashion, while ray-tracking inside a slab is performed parallely. The advantage of the method is that optimization methods which work only for parallel ray casting can be applied. A completely different approach of accelerating perspective ray casting at the expense of image quality or flexibility is image-based rendering. Qu et al. introduced a technique they called keyframeless 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. 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 preprocessing 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. A similar technique was proposed by Serlie et al. [SVvG00]. In recent years, hardware-based ray casting methods have been presented which perform well also for perspective rendering [WS01]. They, however, either lack the needed flexibility or confine rendering to view points outside the volume.

It is clear to understand that it is not necessary to render both foreground and background using the same visualization technique. A combination consisting 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, eg., bone structures or contrasted blood vessels. This reduces the need for tedious manual 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 must be adapted [Hop96] during the virtual investigation, e.g., in order not to waste renderig time during phases when the object is still far away from the view point (see the leftmost image in figure 1). This again requires a lot of preprocessing and virtually prohibits on-the-fly triangulation. For the project described in this paper, a ray casting technique was chosen to be implemented.

This is the outline of this paper: In section 2, preprocessing of segmented objects is discussed. Section 3 deals with the visualization techniques used. The approach for visualizing background objects is described in detail, foreground generation and the fusion of foreground and background are covered briefly. Chapter 4 presents results. A discussion is given in chapter 5.

2. Background Objects

As pointed out already in the introduction, there are two kinds of background objects in virtual endoscopy applications: The first are those which need not be segmented prior to the investigation, because they 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 retrieved from a binary segmentation mask. In order to visualize the surface of such an object, an artificial surface must be constructed around the mask. Gibson [Gib98] presented an approach which constructs a smooth triangle mesh around a binary mask: A vertex is positioned in the center of each cell which is adjacent to at least one voxel inside and at least one voxel outside the binary mask. In each of a series of relaxation steps, each vertex is moved slightly towards the average position of vertices in the neighboring cells, with the constraint that no vertex is allowed to cross a cell boundary. After this smoothing phase, a triangle mesh is created by connecting the resulting vertex positions. This technique is a good choice, if polygon rendering is used to display background objects.

Since in this project a ray casting algorithm is used for visualization, an algorithm which directly manipulates the volume to create an artificial iso-surface around the visualized object is needed - the mask must be *voxelized*. Simple voxelization techniques assign a value v_1 with $v_1 < T$, with T being the iso-value, to the voxels surrounding the mask and a value v_2 with $v_2 > T$ to all voxels belonging to the mask. Such an iso-surface, however, suffers from heavy aliasing, since the binary classification results in a discontinuous inside-outside function with an unbounded frequency spectrum and trilinear interpolation has only limited filtering effect. Thus a considerable fraction of the iso-surface is positioned exactly on the voxel boundaries, visibly revealing the underlying voxel-structure (see figure 2). 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 $\geq T$ and other voxels to values $\leq T$, a constraint similar to the one used by Gibson, as outlined above. 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. If values are confined to be $\geq T+\delta$ or $\leq T-\delta$ with an arbitrary value δ, a large number of filtering iterations yields a quasi-binary classification - voxels near the boundary acquire either the value $T+\delta$ or the value $T-\delta$, 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.

Lakare and Kaufman [LK03] suggest a method that is applicable for transfer function-based direct volume rendering: Intensities of the voxels on and near the boundary of the binary segmentation mask are altered such that a *fuzzy boundary* is created. A similar technique was used for this project: Each voxel near the object boundary is assigned a new value v, with $V_1 \leq v \leq V_2$ and $V_2 - T = T - V_1$ using the following algorithm: A *reference mask* is established by eroding the segmentation mask n times. Since a morphological opening-effect is not desired, 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 its value is set to

$$v_2 - (v_2 - v_1) \cdot \frac{n+1-i}{n+1}$$

All other voxels inside the final reference mask are assigned the value V_2. Then $2n$ dilation operations are performed. Each voxel that is added through dilation is assigned the value

$$v_n - (v_n - v_1) \cdot \frac{d}{2n+1}$$

with d being the distance of the voxel to the nearest voxel V_n in the reference mask and v_n the value of voxel V_n. Nearest voxels can be efficiently found by tracking them with the dilation front. During the first n dilation steps, only voxels

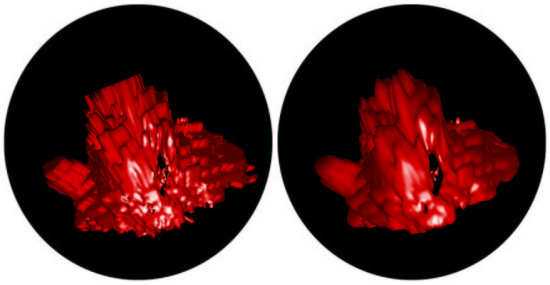

Figure 2: *Left: object imported from binary mask without smoothing; right: smoothed object*

contained in the original segmentation mask are added. This algorithm yields a uniform value degradation from the reference mask into each direction. It can be easily shown that voxels outside the mask always acquire a value $< T$. For voxels inside the mask, $d \leq \sqrt{3n^2}$. If $n = 1$, the voxels inside the mask with maximum d ($d = \sqrt{3n^2}$) acquire values slightly lower than T, yielding a smoothing of sharp edges, all other voxel values inside the mask are $> T$. For higher values of n, object shapes can be significantly altered. Thus, for this project, $n = 1$ was used. Figure 2 shows the result of applying this algorithm to a binary mask.

3. Visualization

For the application outlined in the introduction a visualization technique which keeps rendering speed, flexibility and image quality at a high level, is required. For the simulation of endonasal transphenoidal surgery two different ray casting methods are used, one for the foreground and one for background objects. Therefore, two images are created in each frame. Foreground and background images are then combined using a z-buffer. This chapter focusses on the generation of the background image. 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. In this chapter, first, cell-based first-hit ray casting will be briefly described. The following sections will, in more detail, explain the changes and optimizations that were applied.

3.1. 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 structure, 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 is encountered which contains a part of an iso-surface, 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:

Macro-cell trimming Instead of the whole 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 3.3.

Early scan line termination 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. Unfortunately, early scan line termination is generally less effective when visualizing only background objects, compared to its application for generating the foreground for a virtual endoscopy system (see also section 3.6).

Screen regions A method to speed up the rasterization phase - small rectangular regions of the screen which have already been completely filled, can be skipped by rasterization. This approach is not very effective if only background objects are rendered and was therefore discarded.

The main principle of acceleration in cell-based first-hit ray casting was the reduction of ray traversal lengths. The following sections will introduce some optimization techniques which increase the number of ray steps tracked, but still speed up rendering (sections 3.2, 3.3 and 3.6), by finding compromises satisfying the trade-off between the ray traversal time reduced and the cost associated with reducing it. Sections 3.2, 3.4 and 3.5 explain, how pixel-space coherency can be exploited for further speed-up.

3.2. 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 segmets have to be tracked through the macro-cell. Depending on the current view point, at most three faces of the current macro-cell are visible. 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* which are tracked using Bresenham rasterization to find the projection boundaries for each scan line. Vertices with visibility coefficients of 2 or 3 define the end points of *frontier lines*, which are also rasterized from top to bottom using the Bresenham algorithm. For each scan line the intersection points of the scan line with the silhouette lines define the set of pixels that have to be processed. The position of each pixel relatively to intersections of the scan line with frontier lines indicates, through which macro-cell face the ray enters the macro-cell. The error induced by discretization can result in the algorithm selecting the wrong macro-cell face for pixels which are close to a frontier line. The consequence is that sometimes the ray segment is started outside the macro-cell. In fact, an average of about 7 percent of all ray steps are tracked outside the according macro-cell. The faster intersection test, however, more than outweighs this overhead. The overall rendering time saved amounts to about 8 percent on the average.

3.3. Ray segment concatenation

Often a ray moves along the surface 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. 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 should be entered by a local ray segment. In order to avoid costly computations finding the index of the neighboring macro-cell in the bit-volume, the indices of the three neighboring macro-cells (one for each dimension) should be

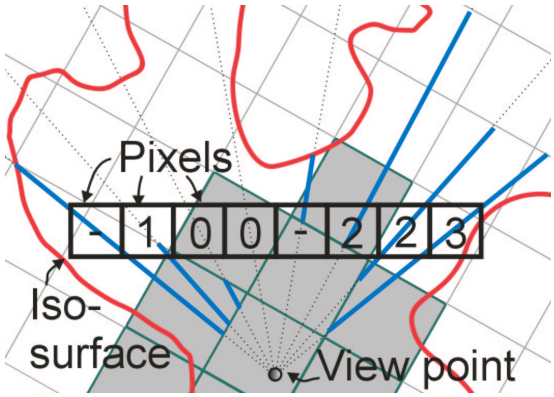

Figure 3: *Ray segment concatenation and* concat *values.*

tracked during the computation of the ray segment and updated every time a macro-cell boundary is crossed. 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 local ray segments having been tracked 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, 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 3 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. Squares filled with gray color correspond to macro-cells which have been processed already. The viewing plane consists of eight pixels, the numbers indicate the current *concat* value of each pixel. 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. If a ray has intersected the iso-surface already, the *concat* value of the according pixel is not important any more. For those pixels, a "-" was written instead.

Ray segment concatenation causes the rasterization algorithm to change slightly: For every 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. This measure, however, does not remove all cases of ray segments tracked twice, the reason for this are rasterization errors: Silhouette lines are discretized - in each scan line the boundary of the macro-cell projection is moved to the nearest pixel center. The consequence is that some pixels close to the boundary are erroneously judged to be inside the

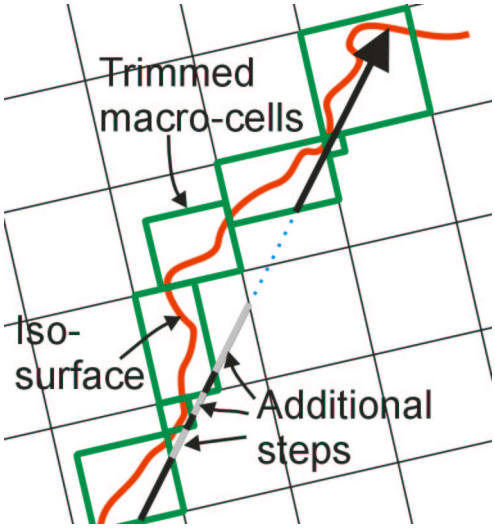

Figure 4: *Increased ray-tracking distance caused by ray segment concatenation*

projection. For these pixels, either a ray segment is tracked, which does not intersect the currently processed macro-cell, or the *concat* value is erroneously decreased, resulting in a local ray segment being redundantly tracked for a second time later on. Another 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 these problems is clearly outweighed by the performance gain brought about by ray segment concatenation: The number of ray segments is reduced by about 57 percent. Despite an increase of tracked ray steps by about 7 percent, rendering time decreases by about 17 percent on the average.

Figure 4 shows the effect of ray segment concatenation on ray tracking distance, with macro-cell trimming having 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.

3.4. Sparse screen sampling

The number auf 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 casted in vain, especially when the eye point is very close to a visualized background object. There are two main reasons for this: Surfaces of background objects very often have a rough shape, surface normals 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 perspectivity, 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 computationaly 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 used for the described project, sampling frequency is determined by two parameters: $skip_{collumn}$, 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. 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 checked to find the real object boundary. Also, for the same reason, scan lines skipped between the previous and current scan lines 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 5. Complete objects whose pojections never intersect the sparse sampling grid, might, however, be lost. In order to keep information loss at a minimum, values $skip_{collumn}$ and $skip_{row}$ can be adapted to the size of the projection of the macro-cell (see also section 3.6). If a macro-cell is projected to a large portion of the screen, one can expect that important

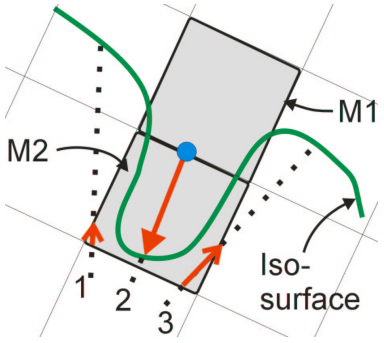

Figure 5: *Tracking ray segments backwards: During processing of macro-cell M2, rays 1 and 3 encounter no intersection, therefore ray 2 is erroneously skipped. Macro-cell M1 is processed after M2. The entry point of ray 2 into M1 is inside the object - the ray must be traced backwards.*

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, $skip_{collumn}$ and $skip_{row}$ wer 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 the average.

3.5. 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. In this project, 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

$$| i_1 - i_2 | < t$$

with the predefined threshold t, 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

```
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 empty ?
      No:
        How many pixels were skipped before?
          1: Apply similarity test for skipped pixel
               Similar: interpolate pixel value
               Not similar: cast ray segment
          More:
            For each skipped pixel:
              Cast ray segment
        Skip 1 pixel
      Yes:
        Skip skip_collum pixels
        For each skipped pixel:
          Reduce value concat by 1
  Has at least 1 intersection been found ?
    Yes:
      How many scan lines were skipped before?
        1: For each pixel in skipped scan line:
             Number of vertical neighbors filled ?
               0: Do nothing
               1: Cast ray segment
               2: Apply similarity test
                    Similar: interpolate pixel value
                    Not similar: cast ray segment
        More: Rasterize skipped scan lines
    No: Skip skip_row scan lines
```

Figure 6: *Pseudo code of the rasterization process*

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 6.

3.6. Adaptive rendering

The algorithm described so far computes all macro-cells in the same way. Only the sampling frequency parameters introduced in section 3.4 differ between macro-cells, depending on the sizes of the macro-cell projections. This section shows some more examples of how this parameter, $p-size$, can be used to increase flexibility of the algorithm in order to maximize performance:

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: For each surface cell inside the macro-cell a gradient must be calculated and projected 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. As a consequence, the application of early scan line termination is confined 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 $p-size$ 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 number of ray segments can effectively be reduced, 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 subvolumes 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 $p-size < T_{macro}$ for a predefined threshold T_macro has been encountered, subvolumes 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.

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 artifacts 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, for the first time, $p-size < T_samp$, the resolution inside the macrocell is coarsened by combining eight cells to one. This can efficiently be done by manipulating the parameters of the fast voxel traversal algorithm.

Applying these adaptive rendering strategies saves about 20 percent of rendering time on the average. The parameter $p-size$ 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, $p-size$ should be normalized with respect to the opening angle. In this project 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 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 chapter) 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.

3.7. Foreground generation

In the project described in this paper, two different ray casting techniques are used. Background objects are visualized using the algorithm described so far. For the foreground, an image order ray casting, basically a low-level-optimized standard first-hit ray casting technique, was applied. There are basically two reasons for that: One reason is that the cavities visualized during virtual endonasal surgery are usually quite small. Therefore rays do not tend to become very long also for image-order ray casting. The second reason is that in order to achieve interactive frame rates for a reasonably sized (a diameter of at least 400 pixels) display on a standard PC, it is usually still necessary to switch to a lower image resolution during phases of interaction. Frame times in image order ray casting scale down approximately with the same factor as the resolution decreases. The object-order technique presented in this paper reduces per-pixel cost whilst generating cost for traversal of the data structure, which, of course, cannot be reduced by rendering to a smaller image. The performance gain achieved by reducing image resolution is therefore not as significant as for an image-order approach.

3.8. Image fusion

In each frame during a virtual endoscopy investigation, two images are generated, one depicting the foreground and one showing background objects of interest. The two images are combined using a weighted average operator, yielding the

	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*

impression of the foreground being semi-transparent. A z-buffer is used to determine those regions of the image where the background is not covered by the foreground. In the project described in this paper, this allows for previewing the effect of punching the sella floor (see the rightmost image of figure 1). The z-buffer can also be used to enhance the impression of semi-transparency: The distance between foreground and background at each pixel determines the level of foreground transparency. This method significantly improves visual depth perception, making rendering more realistic and orientation easier. This effect is illustrated in figure 1. In the leftmost image, background objects (the tumor and blood vessels) are far away from the foreground surface. Therefore they are only faintly visible.

4. 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 4 gives timings acquired during a complete virtual endonasal investigation, 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: 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 figure 1), where background objects cover a large fraction of the screen (the maximum was 78

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 2 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.

5. conclusion

This paper introduced a new algorithm for efficient and flexible visualization of background objects for virtual endoscopy 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. Viewing rays are only tracked within the immediate neighborhood of the visualized objects of interest. This speed-up is achieved at the expense of increased administrational overhead for traversing the data structures. 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.

Acknowledgements

This work has been carried out as part of the basic 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.

[Bar01] BARTROLI A. V.: *Visualization Techniques for Virtual Endoscopy*. PhD thesis, Vienna University of Technology, 2001.

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

[BS99] BARTZ D., SKALEJ M.: Vivendi - a virtual ventricle endoscopy system for virtual medicine. In *Proc. of Eurographics/IEEE TCVG Symposium on Visualization '99* (1999), pp. 155–166.

[BWKG01] BARTROLI A. V., WEGENKITTL R., KÖNIG A., GRÖLLER E.: Perspective projection

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	901,165	4.0	7.2
+ ray segment concatenation	70,460	962,088	4.8	8.7
+ sparse screen sampling	46,268	673,573	6.2	11.1
+ pixel space coherence	31,677	461,143	6.8	12.1
+ adaptive rendering	38,229	447,873	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*

through parallely projected slabs for virtual endoscopy. In *Proc. of SCCG '01- Spring Conference on Computer Graphics* (April 2001), pp. 287–295.

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

[Gib98] GIBSON S.: Constrained elastic surface nets: generating smooth surfaces from binary segmented data. In *Proc. of Medical Image Computation and Computer Assisted Surgery, 1998 (MICCAI '98)* (1998).

[GRF*63] GUIOT G., ROUGERIE J., FOURESTLER A., FOURNIER A., COMOY C., VULMIERE J., GROUX R.: Une nouvelle technique endoscopique: Exploration endoscopiques intracraniennes. *La Presse Medicale*, 71 (1963), 1225–1228.

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

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

[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.

[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.

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

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

[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).

[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.

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

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

[Sra96] SRAMEK M.: *Visualization of volumetric data by ray tracing*. PhD thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, 1996. Indexed at `http://www.iinform.oeaw.ac.at/`.

[SVvG00] SERLIE I., VOS F., VAN GELDER R.: Improved visualization in virtual colonoscopy using image-based rendering. In *Proc. Eurographics/IEEE TCVG Symposium on Visualization (VisSym '00)* (2000).

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

[WS01] WESTERMANN R., SEVENICH B.: Accelerated volume ray-casting using texture mapping. In *Proc. of IEEE Visualization 2001* (2001), pp. 271–Ű278.

[WVH*00] WEGENKITTL R., VILANOVA A., HEGEDÜS B., WAGNER D., FREUND M. C., , GRÖLLER M. E.: Mastering interactive virtual bron-

chioscopy on a low-end PC. In *Proc. of IEEE Visualization 2000* (2000), pp. 461–464.

[YS93] YAGEL R., SHI Z.: Accelerating volume animation by space leaping. In *Proc. of IEEE Visualization '93* (1993), pp. 62–69.