Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2002)
D. Ebert, P. Brunet, I. Navazo (Editors)

Cell-Based First-Hit Ray Casting

A. Neubauer, L. Mroz[†], H. Hauser, and R. Wegenkittl

VRVis Research Center, Vienna, Austria
mailto: {neubauer,hauser,wegenkittl}@vrvis.at

Abstract

Cell-based first-hit ray casting is a new technique for fast perspective volume visualization. This technique, based on the well known ray casting algorithm, performs iso-surfacing and supports interactive threshold adjustment. It is accelerated by the reduction of average ray path lengths to only a few steps per pixel. The volume is divided into cubic sub volumes. Each sub volume that is intersected by an iso-surface is projected to the image plane. A local ray casting step within the sub volume is performed for each pixel covered by the projection. Cell-based first-hit ray casting is perfectly suited whenever fast perspective iso-surfacing is required. This paper describes the basic algorithm, presents possible optimizations and evaluates the performance of the algorithm for one specific application, the post-implantation assessment of endovascular stent placement. It will be shown that the algorithm, though executed on a single processor machine without any hardware acceleration, performs well for view points inside as well as outside the stented blood vessel and significantly outperforms an optimized, yet more conventional ray casting technique.

1. Introduction and Related Work

Stent implantation is a powerful instrument for the treatment of heart function deficiencies. A stent (see figure 1) is a small tubular prosthesis that is inserted into an artery via an endovascular procedure and used, for instance, to enlarge a stenosis, a local narrowing of the arterial lumen. In the past years, the task of efficiently visualizing an inserted stent within the blood vessel for exact assessment of the quality of insertion became a new field of application of three-dimensional medical computer visualization. This task includes the need for interactive virtual angioscopy (virtual endoscopy [1] inside blood vessels) combined with the visualization of the stent (see figure 2) as well as fast visualization with view points outside the vessel to assess the position of the stent with respect to certain landmarks of the body (e.g., heart, lung, bones, etc.).

A feasible way of providing the prerequisites for clean visualizations in the virtual endoscopy application is to define two iso-surfaces by stating two thresholds, one representing the stent boundaries and one representing the vessel walls. During virtual angioscopy investigations, the density value at the view point inside a blood-filled vessel is, due to an injected contrast agent, usually on a higher level than the surroundings of the vessel and lower than the density of the metal stent. Therefore, provided a CT data set is used, both iso-surfaces can be rendered.

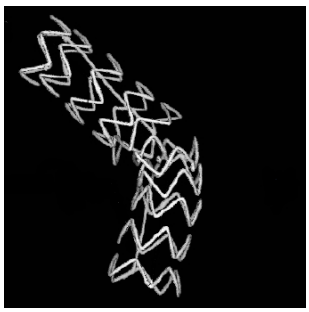

Figure 1: *a stent*

For views from outside the vessel, when the density value at the view point is below the density level of the visualized environment, only one threshold is needed, which visualizes the stent and/or the environment. It is important to provide means of interactive threshold adjustment to allow the user

[†] Tiani Medgraph, mailto: lukas.mroz@tiani.com

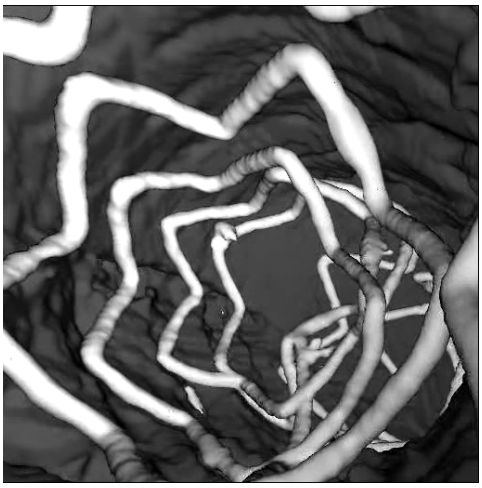

Figure 2: *virtual angioscopy (a virtual flight through a stented blood vessel*

to select the visualized environment and its appearance on the fly. Only then can the user obtain a reasonable understanding of the surroundings of the implanted stent. Traditional surface fitting methods (e.g., marching cubes [2]) have the disadvantage that the costly surface extraction procedure must be repeated whenever the threshold changes, which makes them non applicable for the described purposes. Thus, a decision has to be made between iso-surfacing via direct volume rendering (DVR) and a method which does dynamic surface fitting (SF), extracting only visible parts of the iso-surfaces during each frame and rendering the resulting mesh using polygon rendering hardware.

Many DVR techniques have been proposed [3]. As image fidelity and perspective are of paramount importance in the application described above, the most suitable DVR method seems to be the volume ray casting algorithm [4]. This technique which can easily be adapted to visualize iso-surfaces (*first-hit ray casting*), suffers from the problem that it cannot achieve interactive frame rates without multi-processor hardware. Therefore, much work has been done to accelerate the technique to allow for more interactive rendering. Most of the proposed optimizations rely on one or more of the following principles:

fast empty space traversal Rays often have to travel many steps through the volume until they either hit an iso-surface or exit the volume. This can take a significant portion of rendering time. Therefore, some techniques have been developed, that aim at accelerating traversal through empty regions. Prominent examples are *Space Leaping* [5, 6] and the *Lipschitz method* [7].

data reduction Portions of the data set that do not contribute to the final image are identified and removed before the start of the rendering process. So the rendering process can be speeded up, because rays do not have to travel through non-interesting regions of the data set [8, 9].

improvement of caching behavior Data sets for volume visualization are often large, for instance, between 200 and 1000 slices of 512^2 voxels. In the traditional ray casting algorithm, rays travel, one by one, through the whole data set. Therefore, cache coherency is hard to maintain. By maximally utilizing cached data before they are replaced, thrashing can be minimized [10, 11, 12].

Other principles of acceleration of ray casting like, for instance, simulation of perspective projection [13], usage of frame-to-frame coherency [14] or pixel-space coherency [15], usually offer only reduced output image quality.

Hardware-based approaches have been presented, which speed up ray casting significantly by performing parts of the algorithm in hardware. These techniques, however, comprise flexibility in one way or another. Westermann et al. [16] presented a texture based volume rendering algorithm which finds the first intersection of a ray with the predefined object (based on a transfer function) using texture-mapping hardware. A completely software-based solution, on the other hand, need not depend on transfer functions, but can be extended to visualize also manually segmented objects, which can not be represented sufficiently well by transfer functions (e.g., tumors). If a high level of flexibility is required, algorithms completely executed in software seem to be most applicable.

During the past few years, some surface fitting techniques that execute a surface extraction procedure during each frame have been introduced. They use elaborate visibility determination algorithms which detect those cells inside the data volume at which surface extraction has to be performed. Since the number of those cells is usually quite small compared to the size of the whole volume, surface extraction is not prohibitively expensive [17, 18]. However, those methods either do not yet perform convincingly well or else provide functionality that is limited in one way or the other. The fast technique proposed by Hietala et al. [18], for example, does not allow for interactive threshold adjustment. Furthermore, these techniques also rely on dedicated acceleration hardware. Thus, first-hit ray casting seems to be more applicable for the purposes described above.

Cell-based first-hit ray casting, a new optimized ray casting technique which is introduced in section 2 combines fast empty space traversal, data reduction and improvement of caching behavior to maximize rendering performance while keeping rendering quality at a high level.

2. Cell-Based First-Hit Ray Casting

Cell-based first-hit ray casting is a new and fast ray casting algorithm which is usable for interactive visualization, for example, for the post-implantation assessment of stent

placement. This section gives a detailed description of the algorithm and covers some possible optimizations.

2.1. Algorithm Overview

Cell-based first-hit ray casting performs iso-surfacing. The data volume is divided into cubic sub volumes, so called *macro-cells*. Each macro-cell consists of n^3 cells, where n is usually a number between four and ten. Each macro-cell containing a part of one of the iso-surfaces is projected to the image plane. A rasterization algorithm detects all pixels that are covered by the projection. From each of those pixels that have not yet been assigned a color, a ray is cast through the macro-cell. Sampling starts at the point where the ray intersects the closest face of the macro-cell and stops, when the ray either hits an iso-surface or leaves the macro-cell. Whenever, during its traversal, the ray hits a *boundary cell*, i.e., a cell that contains data values above as well as below one of the specified thresholds, a ray-surface intersection algorithm is applied to see, if and at which position the ray intersects the corresponding iso-surface. If the iso-surface is hit, the gradient vector at the intersection point is calculated using trilinear interpolation. A color value is then calculated from the gradient and assigned to the corresponding pixel.

2.2. The Data Structure

In order to quickly find those macro-cells which contain a part of at least one of the iso-surfaces, a min-max octree is used [19]. A leaf node of the octree represents a macro-cell. For each node of a min-max octree, the minimum and the maximum data value of the sub-space that it represents are stored. Traversal starts at the root of the octree. A child node is processed only, if one of the thresholds is between the minimum and the maximum value of this child node. As soon as a leaf node is reached that passes the min-max check, the corresponding macro-cell is projected and local rays are cast from the covered pixels.

Rays are cut into segments by macro-cell boundaries. During one local ray casting step, only one segment of a ray is processed. Since the first intersection of a ray with an iso-surface that is found, determines the final color of the pixel, it is of utmost importance that ray segments are processed in the correct viewing order. Because perspective projection is used, each pixel is associated with a unique ray direction. However, since all rays that intersect any two sub volumes, intersect them in the same order, a representative ray which intersects all sub volumes can be used to quickly establish a correct order of child node traversal. At each octree traversal step, the only ray that intersects all sub volumes represented by child nodes is the one moving through the center point of the sub volume represented by the parent node. The signs of the components of its direction vector indicate, which child node should be processed first and which should be processed last. The remaining child nodes can be processed in arbitrary order, since no ray can possibly pierce any two of the sub volumes represented by them.

2.3. Local Rays

A local ray is cast for each pixel that is covered by the projection of the currently processed macro-cell. Voxel traversal inside the macro-cell is started by calculating the entry point of the ray into the macro-cell. Then, according to the method presented by Amanatides and Woo [20], all cells that are intersected by the ray inside the macro-cell are detected. For each of those cells it has to be determined, whether it is a boundary cell. If this is the case, a ray-surface intersection test is executed.

2.4. Ray-Surface Intersection

There are some techniques for performing ray-surface intersections, that differ in quality and, especially, in complexity. A very accurate approach was introduced by Lin et al. [21] and by Parker et al. [12]. This intersection algorithm inverts trilinear interpolation. It finds, for a given data value, all points of a given ray that are assigned this data value by trilinear interpolation. If one or more of those points are inside the investigated cell, the first of them along the ray is the intersection point. This technique performs exact intersection, but is computationally very expensive.

A faster algorithm which also yields good results works as follows: The data values at the end points (entry point and exit point) of the path that a ray takes through a boundary cell are calculated. If one data value is greater and one is smaller than the threshold, it is concluded that a ray-surface intersection must exist along the path. This intersection point is estimated using linear interpolation. This process can be repeated recursively: The data value at the estimated intersection point is calculated and, depending on whether this value is greater or smaller than the threshold, a second interpolation is performed either between the entry point and the estimated intersection point or between the estimated intersection point and the exit point. Usually, two such interpolation steps are enough to come sufficiently close to the actual intersection point. This technique produces images whose differences to correct images created by using the exact technique can no more be recognized by the human eye. If only one interpolation step is performed, images are also close to correct, but now and then slightly visible artefacts occur.

Figure 3 displays result images for different ray-surface intersection strategies. The leftmost image was rendered using the exact intersection technique, the image in the middle was rendered using the fast technique (2 interpolations). The rightmost image was created using a trivial method, where simply the center point of the boundary cell is taken as the intersection point. Figure 4 shows two error maps, the left one depicting the differences between the (correct) leftmost image and the center image of figure 3 and the right one

depicting the difference between the leftmost and the rightmost image of figure 3. It can be seen that there are hardly any differences between images created using exact intersections and images created using fast intersection. The trivial intersection test, however, causes clearly visible errors.

Exact intersection tests take about 2.9 times as long as fast intersection tests using two interpolation steps. Therefore, it is a good decision to trade an unneeded amount of accuracy for speed and perform fast intersections.

2.5. Reduction of the Number of Local Rays

Casting a whole ray from the eye point until it either exits the volume or intersects an iso-surface is in general faster than casting many of its segments as local rays, since for each local ray a rather expensive additional initialization process, consisting of the determination of the ray direction, the entry point calculation and the initialization of variables used for cell traversal, has to be performed. Time is only saved, if only few local rays are cast per pixel. *Macro-cell trimming* and *early scan line termination* are two optimization techniques that reduce the number of local rays.

2.5.1. Macro-Cell Trimming

Very often, only a small part of a macro-cell is intersected by an iso-surface. Therefore, to prevent unnecessary calculations, each macro-cell is, before being projected, trimmed to the smallest cuboid containing the macro-cell's portions of the iso-surfaces. Since the projection of this cuboid covers fewer pixels than the projection of the whole macro-cell, the number of local rays is reduced. Also, local rays become shorter, as they only have to move through the final cuboid, not through the complete macro-cell. Macro-cells are trimmed during each frame, not only if the iso-value has been changed. This saves memory and does not induce much additional overhead, since the most costly part, iterating through all cells to find boundary cells, has to be performed anyway to do early scan line termination, an optimization technique described in section 2.5.2. If macro-cell trimming is used, about 28 percent of overall rendering time is saved on average, compared to the pure algorithm which does not require iteration through all cells of surface-containing macro-cells. The time gain, however, is very much dependent on the data set and the position of the view point.

2.5.2. Early Scan Line Termination

Early scan line termination is a technique that reduces the number of local rays cast through a macro-cell by taking into account properties of iso-surface geometry inside this macro-cell. It is based on heuristics and can therefore produce errors. These errors, however, can efficiently be detected and corrected in a post-processing step for each frame.

After a macro-cell has been projected, the set of pixels at which local rays are to be cast is detected by identifying all scan lines that intersect the macro-cell projection and iterating through the pixels of each of those scan lines from one boundary of the projection to the other. Depending on the geometry of the iso-surfaces inside the macro-cell, scan lines can be either rows or columns of pixels and can be processed in either direction (vertical scan lines, for example, can be processed top-down or bottom-up). The reason for this is that, again depending on the geometry of the iso-surfaces, very often, processing of a scan line can be stopped as soon as a local ray through a pixel of this scan line did not intersect any of the iso-surfaces. This technique will be referred to as *early scan line termination*. Here, without limitation of generality, macro-cell projections for which scan lines are processed from left to right will be discussed as an example. Let the *image* of a (macro-)cell be the set of all pixels in the final image, that obtained their colors through ray-surface intersections within that (macro-)cell. Analogously, the *image* of a part of an iso-surface is the collection of all pixels that obtained their colors through ray intersections with that part of an iso-surface. Early scan line termination will lead to errors and should therefore not be applied, if the potentially dropped part of the scan line intersects the image of the current macro-cell. The macro-cell whose projection is depicted in the left image of figure 5 allows early termination of all scan lines. The example scan line drawn in the figure is rasterized only up to the first pixel that remains clear (label "EST"). The scan line drawn in the image on the right should not be terminated early during processing of the illustrated macro-cell, since ray-surface intersections would be missed. There are two possible causes for early scan line termination being disallowed:

The first cause are left-facing surface normals. Parts of the iso-surfaces, which have surface normals that point, when projected to the image plane, towards the left hand side of the screen, confine the applicability of early scan line termination. The reason is that, as can be seen in the example on the right hand side of figure 5, images of parts of an iso-surface facing towards the origin of a scan line are likely to mark an entry point of the scan line into the macro-cell image. Scan line rasterization, of course, should not be terminated before this entry point.

The second possible cause is related to the notion of so-called *dangerous macro-cell faces*. A macro-cell face is referred to as *dangerous*, if it contains the macro-cell vertex which has been projected farthest left and the neighboring macro-cell sharing this face will be processed later than the current macro-cell. If the distance between the two leftmost vertex images in X direction is very small (there is a difference of only a few pixels), only the hidden face shared between the vertices belonging to the two leftmost projection points, is considered dangerous.

The consequence of a dangerous macro-cell face being intersected by an iso-surface is that, although there are no left

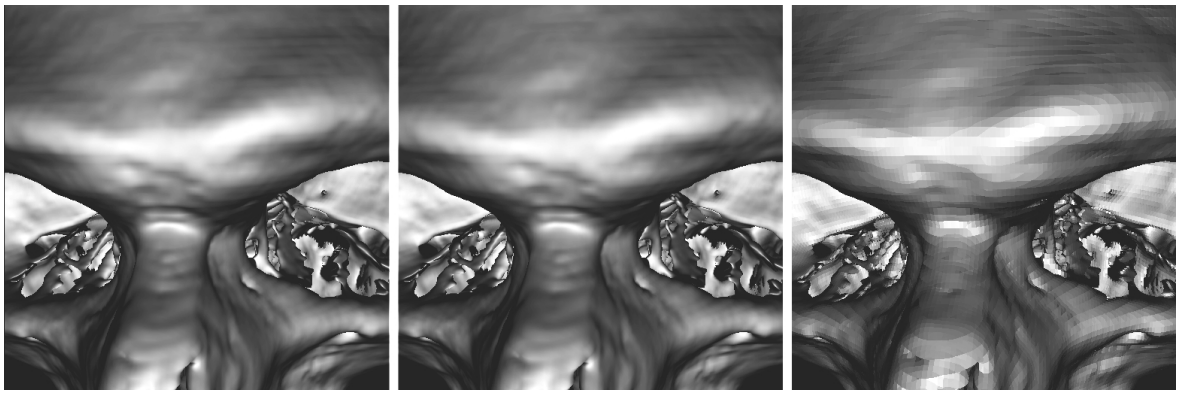

Figure 3: *comparison of three ray-surface intersection techniques. Left: exact intersection, middle: fast intersection, right: trivial intersection (right)*

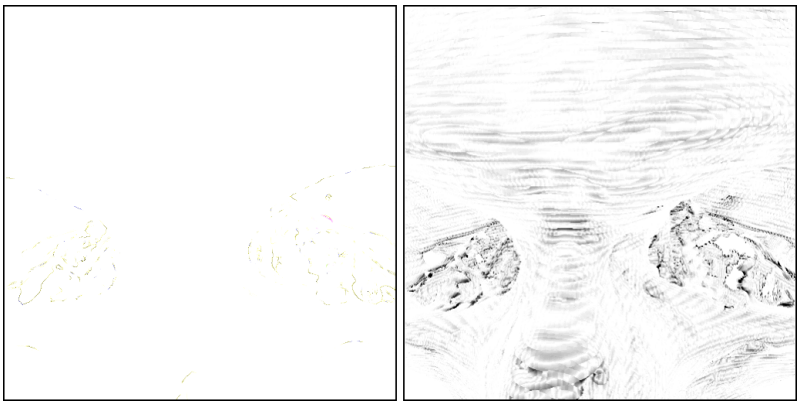

Figure 4: *Differences between exact and fast intersection (left) and between exact and trivial intersection*

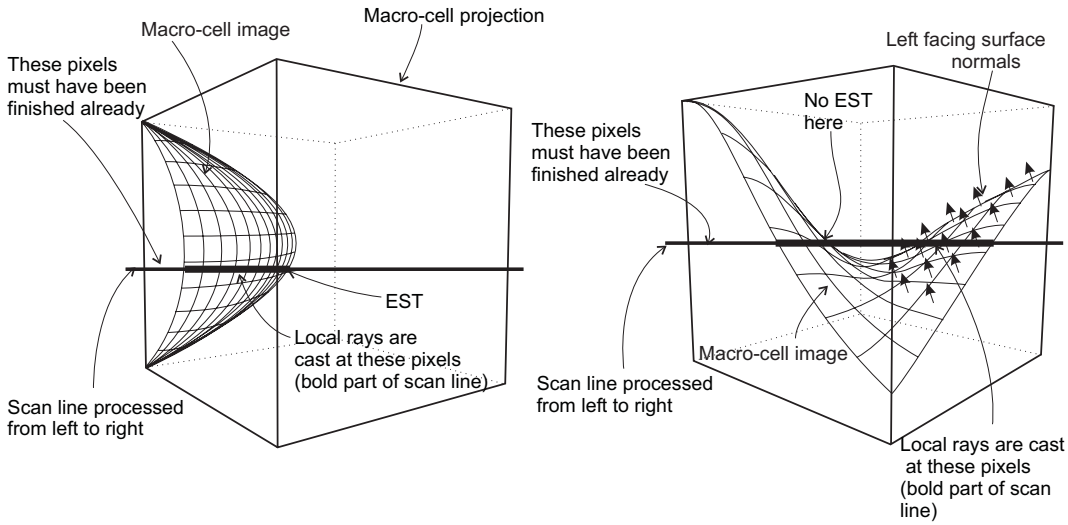

Figure 5: *2 macro-cell projections: left: early scan line termination (EST) is possible for all scan lines; right: an example for a scan line which should not be terminated early due to left facing surface normals*

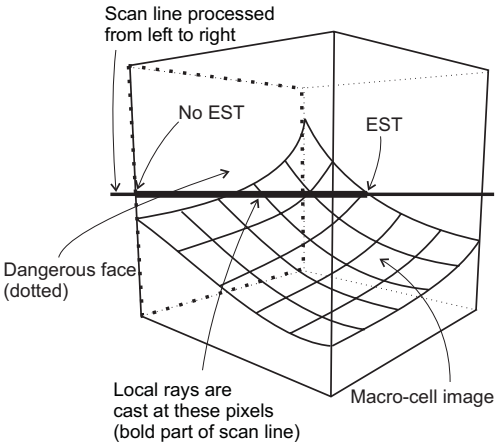

Figure 6: *an example for a dangerous macro-cell face confining the applicability of early scan line termination*

facing surface normals, scan lines might intersect the macro-cell image only after some pixels have remained clear (see figure 6). The reason for this is that rays through those pixels exit the macro-cell through the dangerous face before they intersect an iso-surface. The applicability of early scan line termination is, in this case, confined to the part of the scan line following the first intersection of the scan line with the macro-cell image.

A macro-cell can be checked for both problem cases by calculating a representative surface normal for each boundary cell of the macro-cell and testing its direction. The dot product of each of those representative surface normals and one reference vector for each problem case is calculated. A negative result indicates that scan lines must not be terminated before entering the projection of the corresponding boundary cell, otherwise errors might occur. The reference vector for detecting left facing surface normals is common to all boundary cells. It is parallel to the vector from the upper left corner of the screen in object space to the upper right corner of the screen in object space. Only if a dangerous macro-cell face has been found, boundary cells adjacent to it have to be checked also for the second problem case: First, the scan line, to which the center point of the currently investigated boundary cell is projected, is identified. The vector, which is associated with this scan line in object space and points to the right in image space, is then projected to the plane of the dangerous macro-cell face. The result serves as reference vector for the second problem case. Whenever a boundary cell that prohibits early scan line termination is found, the cell is projected to the image plane. The collection of all those projections yields a *critical region* for the current macro-cell. Early scan line termination can only be executed outside the critical region. Figure 7 shows two examples of critical regions. Critical surface normals which define the critical regions and reference vectors are depicted as well. Curves connecting those pixels at which early scan line termination takes place, are labelled "EST".

Early scan line termination usually saves another 22 percent of overall rendering time on average. The two rightmost images in figure 8 display the numbers of local rays per pixel without, respectively with, the usage of early scan line termination during the computation of the frame shown in the image on the left. The lighter the color of a pixel, the more rays were cast through it. The image in the middle was generated using 2.803 local rays per pixel, the image on the right was generated using 1.143 local rays per pixel.

It has been pointed out that early scan line termination is based on heuristics: Surface normals of parts of the iso-surfaces contained within one boundary cell can vary quite significantly. So, representative normal vectors might describe the iso-surfaces inadequately. It is, for example, possible that a boundary cell contains left facing surface parts, although its representative normal vector does not point leftward. Therefore, critical regions may be too small and errors can occur. Although these errors are very rare (about 1 out of 1000 pixels is erroneous on an average), they are possible and they have a disturbing effect on image quality. They result in holes in the surface, which are clearly visible. Therefore, these holes have to be identified and filled with correct color values after completion of the frame.

2.5.3. Hole Recovery

A pixel that is covered by a hole due to erroneous early scan line termination will be left blank until the end of the frame, because either no other part of an iso-surface is projected to it until the end of the frame, or the ray that is cast through that pixel intersects the same iso-surface one more time, but in the wrong direction (e.g., from high to low density, if the iso-surface represents a stent and a CT data set is used), in which case the pixel is left blank and marked erroneous.

There are two *sequence flags* per pixel indicating whether the pixel has recently been dropped due to horizontal, respectively vertical, early scan line termination. This means that every time that early scan line termination is applied, the appropriate sequence flag has to be set for all empty pixels which are dropped. The bounds of each new sequence of dropped pixels must be marked by setting the sequence flag of the pixel immediately preceding the sequence and the pixel immediately following the sequence to zero. Also, a pixel through which a regular local ray has been cast can, at this time, not be part of an erroneously dropped sequence, thus its sequence flags must be set to zero. It can be concluded from the properties of erroneous pixels described in the previous paragraph, that pixels which are erroneously dropped, will after that never be part of a dropped sequence again. Therefore, after the frame is completed, a sequence of blank pixels with the according sequence flags set, which is bounded on both sides by non blank pixels whose sequence flags are not set, has been dropped erroneously.

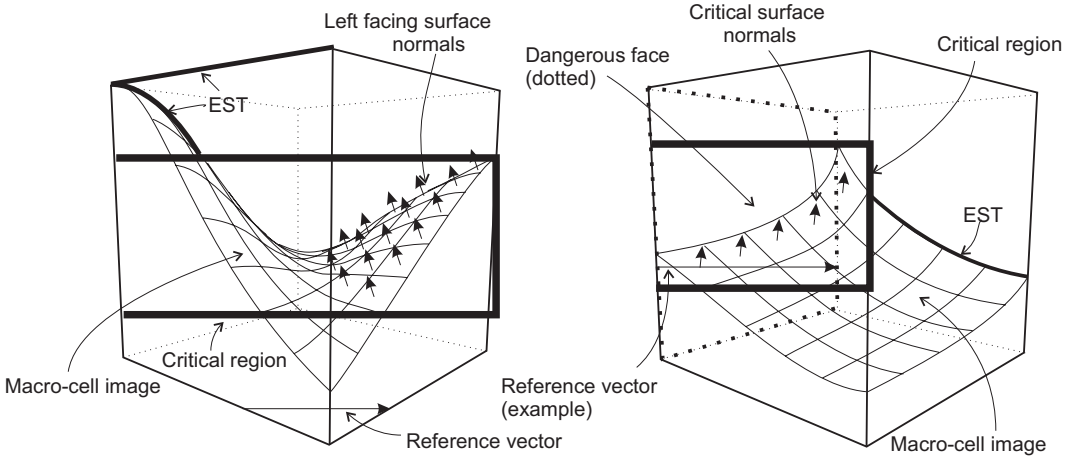

Figure 7: *two examples of the applicability of early scan line termination being confined by critical regions*

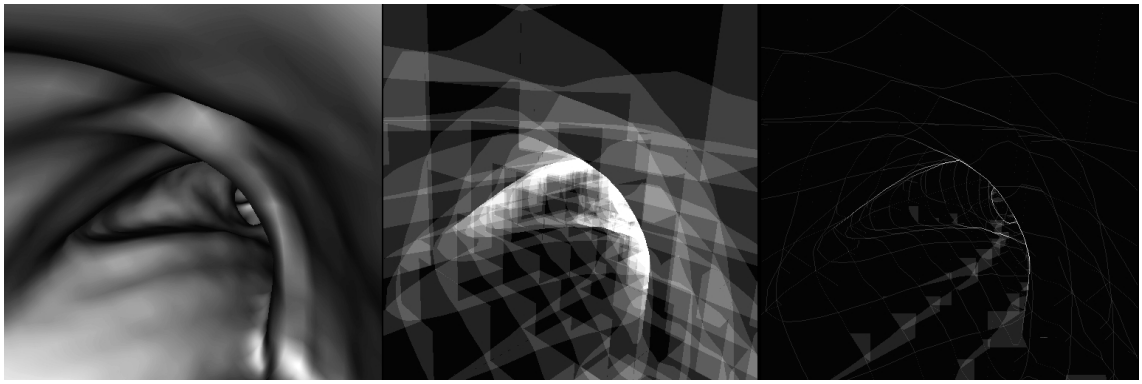

Figure 8: *ray numbers without (middle) and with (right) the usage of early scan line termination*

To fill the holes, rays can be cast from those pixels through the complete volume. As only few pixels will be erroneous, this will mostly take only a small part of the time that was saved by early scan line termination.

2.6. Screen Regions

One problem of cell-based first-hit ray casting is that visibility determination is done on a per pixel basis. So, for instance, after the projection of a macro-cell which is completely hidden behind closer parts of the iso-surfaces, each pixel covered by the projection has to be tested, whether a local ray has to be cast through it, or not. One way to reduce this overhead is the so called *screen regions* method. The image space is divided into equal-sized squares, each consisting of $n \times n$ pixels. For each square the number of pixels which have already been assigned a color, is known. During the rasterization of the projection of a macro-cell, whenever a screen region boundary is crossed, the pixel number of the entered screen region is checked. If the screen region is full, rasterization can move on to the next screen region boundary. This approach saves another 3 percent of overall rendering time on an average.

2.7. Early End of Computation

Intuitively, computation of a frame is ended as soon as the complete octree has been traversed and all macro-cells which include parts of the iso-surfaces, have been processed. However, some time can be saved by terminating the rendering process as soon as all pixels are covered, as this reduces the number of macro-cells having to be projected and tested for dangerous surface normals as well as the number of pixelwise visibility tests. This approach saves another 5 percent of overall rendering time on average.

3. Results

The algorithm described above was implemented and timed on an Intel P4 1900 MHz single processor machine. A

$512 \times 512 \times 266$ data set of a chest was used for the experiments. To test the performance, a reference ray caster which creates images of the same quality as cell-based first-hit ray casting, was implemented and timed on the same machine. This reference program tracks rays from the eye point to those points where they first intersect an iso-surface or leave the data volume. It uses the fast voxel traversal algorithm [20] and performs fast ray-surface intersection tests as described in section 2.4 Additionally, a buffer is used to exploit inter ray coherency. It stores cells that were encountered along the most recent ray, and a flag which indicates, if a portion of an iso-surface is contained there. This abolishes the need, during traversal of the next ray, to check the data values of cells which are already stored in the buffer. This technique accelerates rendering by about 60 percent. Distance coding was not used, because it would require recomputation for each new iso-value. Timings were taken for two scenarios, virtual angioscopy inside the stented aorta and visualization of the stent from outside the chest with various thresholds. Table 1 contains timings obtained during a virtual flight through the aorta. It states the minimum, maximum and average frame times of both techniques for an image resolution of 512×512. It can be seen that cell-based first-hit ray casting is significantly faster in the virtual angioscopy application than more conventional (but still optimized) ray casting. Two frames taken from fly-through animations are depicted in figure 10 <see color section>.

The diagram in figure 9 compares the performances of the two techniques in the second scenario, visualization of the stent from a view point outside the chest. The threshold in the experiment varied between 0 Haunsfield Units (soft tissue, e.g., the lung) and 1100 HU (stent), the image resolution was 512×512. Conventional ray casting is significantly faster than cell-based first-hit ray casting only when soft tissue hides the stent. In this case, distances between the eye point and the ray-surface intersection points are in general very short, thus conventional ray casting does not spend much time traversing empty spaces. Cell-based first-hit ray casting, on the other hand, suffers from the problem that in this case, almost all macro-cells must be processed, and, since the tissue is very fine-structured and there are some "peep holes" into regions farther away, it takes a long time until all pixels have been assigned a color and rendering can be stopped. As the threshold is raised and soft tissue fades out, frame times for cell-based first-hit ray casting decrease, because fewer macro-cells have to be processed, while those for conventional ray casting increase rapidly, because ray lengths increase. With a threshold of 450 HU, when bones and the stent are visible, the frame time of cell-based first-hit ray casting is 1.953 seconds, the one of conventional ray casting is 8.563 seconds (speedup factor 4.38). With a threshold of 1100 HU, when only the stent is visible, the frame time of cell-based first-hit ray casting shrinks to 0.609 seconds, while that of conventional ray casting rises to 12.781 seconds (speedup factor 20.98). Three frames from this experiment are depicted in figure 11 <see color section>.

3.1. Discussion

The most important advantage of cell-based first-hit ray casting are the short paths that rays have to traverse. Rays are tracked only through macro-cells containing a part of the iso-surface. Local rays through parts of macro-cells, which do not contain any boundary cells, are avoided by macro-cell trimming (see section 2.5.1). Early scan line termination eliminates local rays, which are not guaranteed, but highly probable not to find any intersections with an iso-surface (see section 2.5.2). Those techniques result in a very short average ray path length per pixel. Table 2 shows, how cell projection reduces the effort for ray tracking compared to conventional ray casting as well as the effects of the optimizations mentioned above. The values in this table stem from a typical frame of the fly-through animation. A ray step is one iteration of the voxel traversal algorithm. The term *mt* stands for macro-cell trimming and *est* for early scan line termination.

The technique introduced by Parker et al., which also relies on an approach based on *macro-cells*, spends about 60 percent of computation time for ray traversal [12]. Cell-based first-hit ray casting uses only about 25 percent of the frame time for initializing and tracking local rays.

We believe that cell-based first-hit ray casting also manages to exploit cache-coherency to improve performance. As each macro-cell is processed and accessed only once during the calculation of a frame, the phenomenon of thrashing can not occur. However, the algorithm trades computational complexity against memory access, thus the CPU has more time to fill its caches and the system might not be memory bandwidth bound anymore. So, whether the higher level of caching coherency does really yield significant performance improvements will be subject to further research.

The disadvantage of cell-based first-hit ray casting is that some parts of the algorithm entail costs that are constant with respect to image resolution and depend on the size of the data set and/or macro-cell size (see following paragraph). Examples of such parts of the algorithm are octree traversal, macro-cell trimming and macro-cell projection (see section 2.1). These costs must be compensated by exploiting the short average ray paths to yield smaller per-pixel overhead. Therefore, cell-based first-hit ray casting performs poorly compared to more conventional ray casting when rendering low-resolution previews. To render lower-quality previews, either some conventional ray casting scheme or a different preview strategy, like the *progressive rendering* method that was used in the AlVis [22] project, should be employed.

Frame times can depend very much on the macro-cell size. Ideal macro-cell sizes differ from data set to data set

algorithm used	minimum frame time	maximum frame time	average frame time
cell-based first-hit ray casting	1.032	1.289	1.228
conventional ray casting	1.783	2.271	2.119

Table 1: *frame times in seconds for virtual angioscopy*

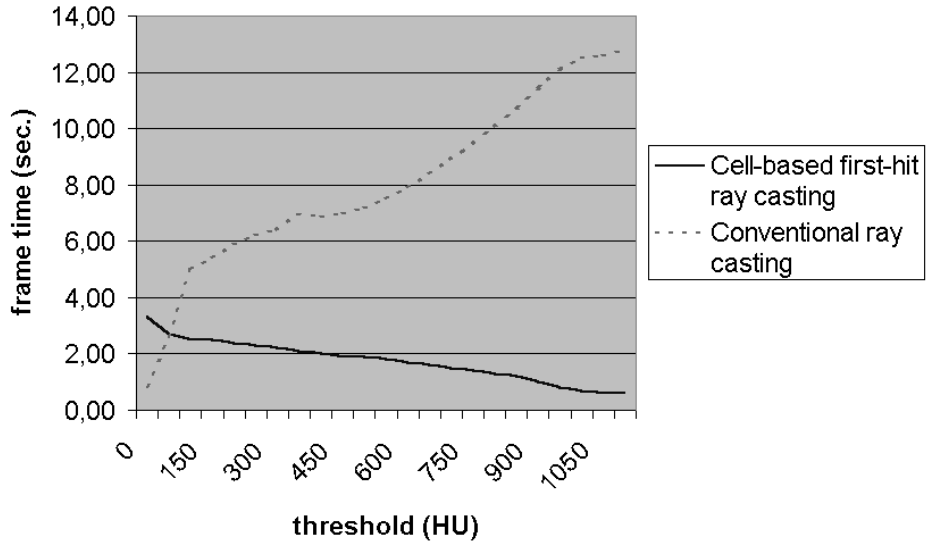

Figure 9: *frame times in seconds for various thresholds*

and even from frame to frame. Both big and small macro-cells have their advantages and disadvantages. A small macro-cell size results in a large number of macro-cells and therefore an increase in the time needed for octree traversal and projection of macro-cells. As there are more macro-cell boundaries, rays are split up into more ray segments. This results in an increase in the number of local rays and thus in an increase in the number of expensive ray initialization steps for rays which travel along an iso-surface without intersecting it for a long time. On the other hand, small macro-cell sizes can lead to more significant reductions of local ray numbers through early scan line termination, as small critical regions become more likely. Also, smaller macro-cells yield shorter local rays, boosting rendering for especially those pixels at which only one local ray is cast.

3.2. Conclusion

Cell-based first-hit ray casting, a new algorithm which proved to be a feasible and fast visualization technique for perspective volume visualization, has been presented in this paper. It was successfully tested in the application of post-implantation assessment of stent placement. It has been shown that cell-based first-hit ray casting manages to outperform conventional but still optimized ray casting quite significantly, for view points both inside and outside the stented blood vessel.

More result images and animations are available on http://www.VRVis.at/vis/resources/DA-ANeubauer/. A more detailed description of the algorithm can be found in the master's thesis *Cell-Based First-Hit Ray Casting* [23].

Acknowledgments

Thanks to M. E. Gröller for his support and to Markus Hadwiger for valuable hints. This work has been done as part of the basic research on visualization (http://www.VRVis.at/vis/) at the VRVis research center in Vienna, Austria (http://www.VRVis.at). Thanks also to Tiani Medgraph (http://www.tiani.com) for providing medical data sets.

References

1. R. A. Robb, "Virtual (computed) endoscopy: Development and evaluation using the visible human dataset",

algorithm used	no of rays	no of rays per pixel	no of ray steps per pixel
conventional ray casting	262,144	1	35.494
cell-based first-hit ray casting	1,147,347	4.376	15.626
cell-based first-hit ray casting with mt,est	396,808	1.364	3.541

Table 2: *contribution of optimization techniques to ray path reduction*

in *Proc. of Visible Human Project Conference '95*, pp. 221–230, (1995).

2. W. E. Lorensen and H. E. Cline, "Marching cubes: a high resolution 3d surface construction algorithm", in *Proc. of SIGGRAPH'87*, pp. 163–169, (1987).
3. M. Meissner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis, "A practical evaluation of popular volume rendering algorithms", in *Proc. of Volume Visualization 2000*, pp. 81–88, (2000).
4. M. Levoy, "Display of surfaces from volume data", *IEEE Computer Graphics and Applications*, **8**(5), pp. 29–37 (1988).
5. D. Cohen and Z. Shefer, "Proximity clouds - an accelerated technique for 3D grid - traversal", *The Visual Computer*, **11**, pp. 27–38 (1994).
6. J. Freund and K. Sloan, "Accelerated volume rendering using homogeneous region encoding", in *Proc. of IEEE Visualization '97*, pp. 191–196, (1997).
7. B. T. Stander and J. C. Hart, "A Lipschitz method for accelerated volume rendering", in *Proc. of IEEE Visualization '95*, pp. 107–114, (1995).
8. W. Li, M. Wan, B. Chen, and A. Kaufman, "Virtual colonoscopy powered by VolumePro." Indexed at http://www.cs.sunysb.edu/.
9. H. Shen, C. D. Hansen, Y. Livnat, and C. R. Johnson, "Isosurfacing in span space with utmost efficiency", in *Proc. of IEEE Visualization '96*, pp. 287–294, (1996).
10. A. Law and R. Yagel, "Multi-frame thrashless ray casting with advancing ray-front", in *Proc. of Graphics Interfaces 1996*, pp. 70–77, (May 1996).
11. K. L. Novins, F. X. Sillion, and D. P. Greenberg, "An efficient method for volume rendering using perspective projection", *Computer Graphics*, **24**(5), pp. 95–100 (1990).
12. S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P. Sloan, "Interactive ray tracing for isosurface rendering", in *Proc. of IEEE Visualization '98*, pp. 233–238, (1998).
13. A. V. Bartroli, R. Wegenkittl, A. König, and E. Gröller, "Perspective projection through parallely projected slabs for virtual endoscopy", in *Proc. of SCCG '01-Spring Conference on Computer Graphics*, pp. 287–295, (April 2001).
14. H. Qu, M. Wan, J. Qin, and A. Kaufman, "Image based rendering with stable frame rates", in *Proc. of IEEE Visualization 2000*, pp. 251–258, (2000).
15. M. Levoy, "Volume rendering by adaptive refinement", *The Visual Computer*, **6**(1), pp. 2–7 (1990).
16. R. Westermann and B. Sevenich, "Accelerated volume ray-casting using texture mapping", in *Proc. of IEEE Visualization 2001*, pp. 271–278, (2001).
17. J. Gao and H. Shen, "Parallel view-dependent isosurface extraction using multi-pass occlusion culling", in *proc. of IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, (October 2001).
18. R. Hietala and J. Oikarinen, "A visibility determination algorithm for interactive virtual endoscopy", in *Proc. of IEEE Visualization 2000*, pp. 29–36, (2000).
19. J. Wilhelms and A. V. Gelder, "Octrees for faster isosurface generation (extended abstract)", *Computer Graphics*, **24**(5), pp. 57–62 (1990).
20. J. Amanatides and A. Woo, "A fast voxel traversal algorithm for ray tracing", in *Proc. of Eurographics '87*, pp. 3–10, (1987).
21. C. Lin and Y. Ching, "An efficient volume rendering algorithm with an analytic approach", *The Visual Computer*, **12**(10), pp. 515–526 (1996).
22. A. Neubauer and A. Kanitsar, "AlVis - meeting the tremendous requirements arising with the visualization of aluminium foam samples investigated by high resolution industrial CT - modalities", in *Proc. of CESCG 2000- Central European Seminar on Computer Graphics*, pp. 229–242, (May 2000).
23. A. Neubauer, "Cell-based first-hit ray casting", Master's thesis, VRVis Research Center, (2001). Indexed at http://www.VRVis.at/.