EUROGRAPHICS 2000 / M. Gross and F.R.A. Hopgood
(Guest Editors)

Volume 19 (2000), Number 3

Interactive High-Quality Maximum Intensity Projection

Lukas Mroz, Helwig Hauser †, Eduard Gröller ‡

Abstract

Maximum Intensity Projection *(MIP) is a volume rendering technique which is used to visualize high-intensity structures within volumetric data. At each pixel the highest data value, which is encountered along a corresponding viewing ray is depicted. MIP is, for example, commonly used to extract vascular structures from medical data sets (angiography). Due to lack of depth information in MIP images, animation or interactive variation of viewing parameters is frequently used for investigation. Up to now no MIP algorithms exist which are of both interactive speed and high quality. In this paper we present a high-quality MIP algorithm (trilinear interpolation within cells), which is up to 50 times faster than brute-force MIP and at least 20 times faster than comparable optimized techniques. This speed-up is accomplished by using an alternative storage scheme for volume cells (sorted by value) and by removing cells which do not contribute to any MIP projection (regardless of the viewing direction) in a preprocessing step. Also, a fast maximum estimation within cells is used to further speed up the algorithm.*

1. Introduction

The ability to depict blood vessels is of great importance for many medical imaging applications (angiography). CT and MRI scanners can be used to obtain volumetric data sets, which allow the extraction of vascular structures. Especially data originating from MRI, which is most frequently used for this purpose, exhibits some properties which make the application of other volume visualization techniques like ray casting[4] or iso-surface extraction[6] difficult. MRI data sets, for example, contain a significant amount of noise. Inhomogeneities in the sampled data make it difficult to extract surfaces of objects by specifying a single iso-value. In addition, vascular structures and especially thin vessels cover a wide range of data values, which makes the extraction by conventional techniques also difficult.

Maximum intensity projection (MIP) exploits the fact, that within angiography data sets the data values of vascular structures are higher than the values of the surrounding tissue. By depicting the maximum data value seen through each pixel, the structure of the vessels contained in the data is captured. A straight-forward method for calculating MIP is to perform ray casting and search for the maximum sample value along the ray instead of the usual opacity and color composition. In contrast to direct volume rendering, no early ray termination is possible, making standard MIP even more expensive.

Usually, MIP contains no shading information, depth and occlusion information is lost. Structures with higher data value lying behind a lower valued object appear to be in front of it. The most common way to ease the interpretation of such images is to animate or interactively change the viewpoint while viewing. This can be achieved using one of the interactive MIP techniques[1, 3, 7] which - up to now - were not able to generate high-quality images. For performance reasons these techniques perform no resampling of the original data values during maximum evaluation, thus introducing aliasing and producing images of moderate quality (Figure 1). For exact depiction of even small features there is need for algorithms which produce high-quality MIP in real-time. In contrast to interactive MIP techniques which perform just nearest neighbor interpolation, more accurate evaluation of ray maxima is required for the generation of high-quality MIP. At the cost of significantly longer computation times, this allows to create much more detailed images and accurate animations.

Depending on the quality requirements of the resulting image and the desired performance, different strategies for finding the maximum value along a ray can be used:

† former name: Helwig Löffelmann

‡ Institute of Computer Graphics, Vienna University of Technology, Karlsplatz 13/186/2, A-1040 Vienna, Austria. email:{mroz, hauser, groeller}@cg.tuwien.ac.at

(a) (b)

Figure 1: *MIP with a) shear-warp projection with nearest neighbor interpolation b) ray casting and trilinear interpolation.*

- **Analytical solution:** Usually data values within a cell are reconstructed using trilinear interpolation of the data values at the cell vertices. For each data cell, which is intersected by the ray, the maximum value encountered by the ray is calculated analytically. For trilinear interpolation within cells this means that the maximum of a cubic polynomial must be determined. This is the most accurate but also computationally most expensive method[9].
- **Sampling and interpolation:** As usually done for ray casting, data values are sampled along the ray using trilinear interpolation (See Figure 6a). The cost of this approach depends on the resampling step size along the ray. Depending on how many interpolations, which do not affect the result, can be avoided[9, 10], acceleration can be gained up to a certain limit.
- **Nearest neighbor interpolation:** Values of the data samples closest to the ray are taken for maximum estimation. In combination with discrete ray traversal this is the fastest method. As no interpolation is performed, the voxel structure is visible in the resulting image as aliasing[1] (See Figures 1a, 6b).

Recent algorithms for MIP employ a set of approaches for speeding up the rendering:

- **Optimization of ray traversal and interpolation:** Sakas et al.[9] evaluate cell maxima only if the maximum value of the examined cell is larger than the ray-maximum calculated so far. For additional speedup they use integer arithmetics for ray traversal and a cache-coherent volume storage scheme. Zuiderveld et al.[10] apply a similar technique to avoid trilinear interpolations. In addition, cells containing only background noise (usually rather low values) are not interpolated. For further speedup a low-resolution image containing lower-bound estimations of the maximum of pixel clusters is generated before the main rendering step. Cells with values below this bound can be skipped when the final image is generated. Finally a distance-volume is used to skip empty spaces. A careful definition of "empty space" is required to avoid that small, low valued structures are missed.
- **Use of graphics hardware:** Heidrich et al.[3] use conventional polygon rendering hardware to simulate MIP. Several iso-surfaces for different threshold values are extracted from the data set. Before rendering, the geometry is transformed in a way, that the depth of a polygon corresponds to the data value of its iso-surface. MIP is approximated by displaying the z-buffer as a range of gray values. Recent versions of SGI OpenGL include blending modes for maximum evaluation and the SGI Volumizer API includes a MIP option. However the advantages of hardware-acceleration for high-quality MIP are only available for upper range graphics workstations. Recently the VolumePro board[8] became available as a purely hardware-based solution for volume rendering and also MIP. For generating MIP of a quality which is comparable to our approach, 4 times super-sampling has to be employed, reducing the frame rates of the VolumePro board to approximately 4 fps for a 128^3 volume.
- **Splatting and shear warp factorization:** Several approaches[1, 2] exploit the advantages of shear-warp rendering[5] to speed up MIP. Cai et al.[1] use an intermediate "worksheet" for compositing interpolated intensity contributions of voxels for projection of a single slice of the volume. The worksheet is then combined with the shear image to obtain the maxima. Several splatting modes with different speed / quality tradeoffs are available. Run-length encoding and sorting of the encoded segments by value are used to achieve further speedup.
- **Elimination of irrelevant voxels and alternative storage schemes:** In a previous paper we showed, that many voxels of a volumetric data set do not contribute to any MIP[7]. Significant speedup is gained by identifying them before rendering and excluding them from image generation. To avoid overhead for skipping them during rendering, voxels which might contribute to a MIP image are stored in a list and sorted according to their data value. By processing this list from low to high intensity voxels and

projecting them using a computationally inexpensive projection (template-based or shear-warp parallel projection), real-time MIP is achieved (See Figure 6c). One purpose of this paper is to extend this idea to high-quality MIP using trilinear interpolation.

Two major limitations to the performance of state-of-the-art software based high-quality MIP can be identified. First, although regions of the volume which do not contain meaningful data can be identified in advance and skipped during rendering using distance volumes or similar structures[10], the overhead associated with evaluating the additional information and stepping over these cells significantly limits the possible speedup of volume traversal. Secondly, despite of space leaping approaches, the number of interpolations which are actually performed is still far from optimum. As the volume is traversed in a spatially ordered manner along viewing rays, local maxima are usually encountered and evaluated before the global ray maximum is reached. Moreover, lots of unnecessary evaluations are performed on the rising slopes of data value which precede a maximum.

In this paper we present a new algorithm for the generation of high-quality MIP (parallel projection) which is approximately one to two orders of magnitude faster than other software-based approaches with comparable quality. In Section 2 we present a preprocessing scheme, which can (but not necessarily has to) be applied to identify and exclude non-contributing cells from the volume. To maximize the amount of cells, which do not contribute to any MIP, 12 sets of cells are generated, corresponding to 12 clusters of viewing directions. Usually, more than two thirds of all cells can be eliminated from the data completely, no longer causing any overhead to identify them later and skip over them. This is achieved by resorting the cells according to their maximum value (Section 3). As the spatial order of processing cells is not relevant for maximum evaluation, cells containing high data values are evaluated first. This reduces the number of evaluations required for MIP significantly. To avoid even more of the relatively expensive trilinear interpolations, we use a fast method for estimating the maximum value along a ray-cell intersection (Section 4). Using these techniques, we are able to greatly reduce the number of trilinear interpolations required per image pixel, achieving interactive high-quality MIP.

2. Preprocessing of Volume Data

The methods described in the following sections are based on an interpretation of the data set as a regular grid of cells, each one defined by eight data samples of the volume located at the cell's vertices. Within cells trilinear interpolation is assumed. For image generation continuous rays are shot through the pixels of the image, allowing arbitrary image sizes as well as oversampling.

2.1. Motivation

Although time used for volume traversal is more significant as a portion of overall rendering time for computationally inexpensive maximum evaluation and projection methods, identifying and skipping not interesting regions also lowers the number of candidates for a costly maximum evaluation. For trilinear interpolation within cells the cell maximum is always located at one of the vertices. Besides skipping pre-identified empty regions of the volume using distance volumes, another simple way to increase efficiency is to perform maximum evaluations only within cells with a cell maximum C_{max} greater than the current ray maximum. This technique avoids the evaluation of cells reached by rays after processing a (local) maximum. If in addition a good lower-bound estimation of the ray maximum can be obtained before rendering, also cells encountered before the maximum can be skipped, if their maximum is lower than the lower-bound estimation.

Although these methods reduce the number of required evaluations significantly, the lower-bound estimation has to be performed for each new viewing direction and much time is spent during rendering on identifying and then skipping cells and empty regions.

To increase time savings during rendering, cells which do not contribute to any MIP should be identified and removed during a preprocessing step.

2.2. Cell Removal

A cell C does not contribute to MIP from any viewing direction (and therefore should not to be considered for rendering), if all rays passing through it collect a higher value by passing through other cells D either before or after C is processed. Our first approach is to investigate direct neighbors D_i of a cell C only. C does not contribute to any ray through it if $\forall i | D_{imin} \geq C_{max}$. As we assume continuous rays to be traced through the volume, only face-connected neighbors of C have to be considered (A ray entering C through an edge or vertex at least "touches" some face-connected neighbors).

Applying such a unified relevance detection for cell elimination, which considers the whole domain of viewing directions, is very ineffective and leads to low cell removal rates. Significantly better results can be achieved if several distinct clusters of similar viewing directions are distinguished. For rendering, the set of cells corresponding to the cluster which contains the current viewing-direction is selected and used for MIP. To minimize the number of neighbors which have to be checked for elimination, a decomposition of all viewing directions into 24 clusters is performed first. Each cluster of viewing directions corresponds to a quarter-face of the directional cube as depicted in Figure 2b. Considering just viewing directions out of one cluster a cell's relevance to a MIP depends on just three neighbors (Figure 2c) instead of

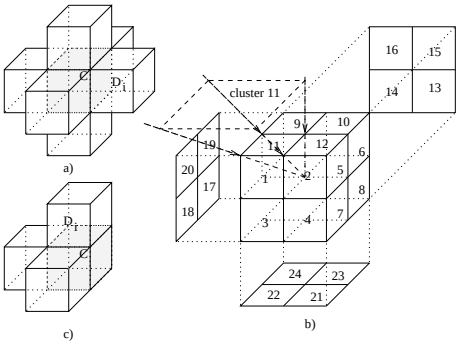


Figure 2: *a) relevance of cell C depends on values of 6 face-connected neighbors D_i. b) decomposition of the viewing-domain into 24 (12) clusters of viewing directions. c) for any viewing direction within cluster 11 relevance of C depends on the values of at most 3 direct neighbors*

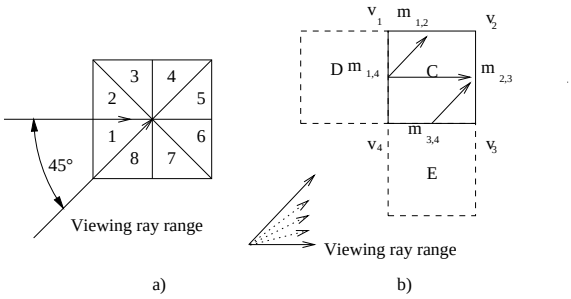


Figure 3: *a) Viewing domain decomposition in 2D. b) face-maximum propagation for cluster 1*

six as in the case of viewpoint-independent preprocessing (Figure 2a).

As the direction of ray traversal is not relevant for MIP and a MIP generated from a certain viewing direction produces exactly the same image as a MIP from the opposite direction, sets of possibly contributing cells have only to be calculated for 12 of the 24 clusters of viewing directions. Furthermore, as can be seen in Figures 2b and c, three clusters of viewing directions around each corner of the directional cube result in the same set of direct neighbors on which a cell's relevance depends (for example clusters 1, 11 and 20). Unfortunately, the clusters differ in the way in which the estimations for rays entering through cell faces combine to the estimations for exiting faces. As our removal algorithm uses face estimations for determining non-contributing cells, the clusters can not be combined without sacrificing removal efficiency.

To achieve effective cell elimination, not only the influence of direct neighbors, but also the influence of more distant cells on rays has to be investigated. This can be done by applying a simple two-pass scheme for each cluster of viewing directions. For reasons of simplicity, the procedure will be explained in 2D. The extension to 3D is straight-forward.

For cell removal in 2D, the viewing domain is decomposited into 8 clusters, each one covering a range of viewing directions spanning 45 degrees (Figure 3a). Rays within cluster 1 can enter cell C only through edge $\overline{v_1v_4}$ or $\overline{v_4v_3}$. Considering just the values at the cell's vertices, cell C has no influence on the maximum of rays through $\overline{v_1v_4}$, if $min(v_1, v_4) \geq max(v_2, v_3)$. In this case, the maximum contribution of the cell to any ray entering through $\overline{v_1v_4}$ and leaving through $\overline{v_1v_2}$ or $\overline{v_2v_3}$ is located on the edge $\overline{v_1v_4}$. As this edge is shared with cell D which is traversed by the rays earlier, C does not contribute to the rays. Similarly, the cell has no influence on rays through $\overline{v_4v_3}$ if $min(v_3, v_4) \geq max(v_1, v_2)$ due to the contribution of cell E. The condition for $\overline{v_4v_3}$ has to consider the influence of v_1, as a large data value at v_1 may influence the data gradient within the cell in a way that rays through $\overline{v_4v_3}$ obtain a larger value within the cell than at $\overline{v_4v_3}$ or $\overline{v_2v_3}$.

Preprocessing the volume in a spatially consecutive order, the influence of cells on ray maxima can be propagated with an advancing front approach. In the example in Figure 3 cells D and E are processed before cell C is reached. For both of them, lower-bound estimations of the maximum for rays which leave the cells have been calculated (for this cluster, rays leave cells either through face $\overline{v_2v_3}$ or through face $\overline{v_1v_2}$). This means, that at the time C is processed, lower-bound estimations $m_{1,4}$ for rays entering through v_1v_4 and $m_{3,4}$ for rays entering through v_3v_4 are available, which already include the influence of more distant (preceding) cells and the influence of the edges themselves ($min(v_1, v_4)$ and $min(v_3, v_4)$). Thus, the revised test for the irrelevance of cell C is $m_{1,4} \geq max(v_2, v_3)$ and $m_{3,4} \geq max(v_1, v_2)$. If both conditions are true, C is removed and never ever considered for MIP anymore. After classifying the cell, the lower-bound estimations for the maxima of rays leaving the cell have to be computed. As for the investigated cluster of viewing directions only rays which have entered the cell through $\overline{v_1v_4}$ can leave through $\overline{v_1v_2}$, the estimation for rays through $\overline{v_1v_2}$ is $m_{1,2} = max(min(v_1, v_2), m_{1,4})$. As rays entering through both, $\overline{v_1v_4}$ and $\overline{v_3v_4}$ can leave through $\overline{v_2v_3}$, the estimation for $v_2v_3 = max(min(v_2, v_3), min(m_{1,4}, m_{3,4}))$.

While the first sweep allows to identify and remove low-valued cells located behind higher-valued parts of the volume, a second sweep in the opposite direction is required to propagate the influence of high-valued cells to cells reached earlier by the rays of this cluster. The second sweep is identical with the first sweep with the exception of the inverted orientation of the rays and thus an inverted volume scan order. Cells which have been removed during the first sweep do not influence ray values during the second sweep. Considering also cells removed during the first sweep would result in mutual elimination of cells and lead to holes in the volume.

The two-pass method presented above can be extended to

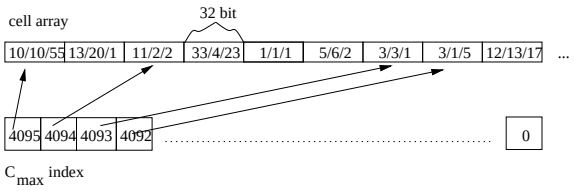


Figure 4: *Cell array storage scheme: all cells are sorted according to C_{max} in descending order. Their position in space is stored in an array (4 bytes per cell). An additional index points to the first cell in the array for each possible value of C_{max}. All cells with the same C_{max} are sub-sorted according to C_{min} in descending order.*

3D in a straight-forward way. For a decomposition of the viewing domain into 24 (12) clusters, rays may enter and leave cells through three faces for each cluster. Lower-bound estimations have to be propagated for faces instead of edges.

2.3. Noise Compensation

To even more increase the efficiency of removal and compensate for noise which is usually present in MRI data sets, the removal process can be modified to remove also cells which violate the exact criteria by a factor which does not exceed a user specified threshold (removal tolerance). After the preprocessing with a 1% tolerance, on average only about 30% of all cells remain as possibly contributing for a single view-set. For detailed results please refer to Section 5 and Table 3.

3. Cell Storage

In the following, we will use $(x,y,z) = (min(x_i), min(y_i), min(z_i))$, (x_i,y_i,z_i) being the vertex coordinates, as reference coordinates of a cell.

As the order in which cells are processed is not relevant for MIP, we are able to abandon the usual storage scheme for volumes, which is a three dimensional array. We use, instead, a storage scheme which is well suited for omitting irrelevant cells during rendering[7]. Cells are sorted according to descending maximum values within a 1D cell array. For every cell its reference coordinates are stored using 4 Bytes – packing 3 indices (x,y,z) allows to store volumes up to a size of 2048x2048x1024 cells. Assuming data values to be represented by 2 bytes, this alternative storage scheme requires twice the amount of memory (without any cell removal). Using this alternative storage scheme it is trivial to start MIP with high-valued cells. Speed up factors of 10-20 compared to other optimized high-quality MIP approaches are gained by just this part of our approach in combination with the usual optimizations like evaluating only cells with $C_{max} >$ ray max and noise skipping (See Table 1, row 3).

Although sorting all cells within a set by C_{min} would most effectively reduce the number of interpolations required, sorting them by C_{max} has several advantages. First, C_{max} is implicitly encoded into the position of a cell in the array by using an additional array of indices which contains, for each possible value of C_{max}, the index of the first cell in the cell array with the particular value of C_{max} (see Figure 4). The implicit encoding of C_{max} allows to access it in an extremely efficient way for testing a cell's relevance during projection. Within a group of cells with the same C_{max} sub-sorting is done according to descending C_{min}. Due to the small range of different values, sorting by C_{max} and C_{min} can be done using fast histogram-based sorting (complexity $O(N)$). During projection, cells with high C_{max} and C_{min} are processed first.

For display of medical images usually a remapping of data values to gray levels (windowing) is applied to enhance the contrast for a specific range of data values which is of utmost interest to the viewer. A window is defined by a *center c* and a *width* value *w*, and defines a mapping of data-values below $c - w/2$ to black, values above $c + w/2$ to white, and values between $c - w/2$ and $c + w/2$ to a uniform ramp of gray levels. If cells are sorted by C_{max} and rendering is started with highest values of C_{max}, rendering can be stopped after reaching the first cell with C_{max} mapped to black. For realistic window definitions as used by medical doctors this can significantly speed up the rendering (up to several times faster).

Besides the fast computation of MIP due to re-sorting, another big advantage is gained: progressive refinement is achieved automatically, as cells which are most relevant to MIP are projected first. Projection can be stopped any time - the result will always be optimal for the given time-constraints. Also, computation of cheap previews is simple. Since interaction is crucial for using MIP, this advantage is also very important for practical use.

In the following a rough comparison between the traditional way of volume storage and our cells array is given: Storing the position instead of data values doubles the amount of memory required. As roughly 30% of all cells remain after the preprocessing step (cell removal), the amount of memory required is about 0.6 * original volume size (per cluster of viewing directions). For 12 clusters this results in an approximately sevenfold increase in memory size compared with the original data set. Considering data sets from medical applications and regular hardware resources, the storage requirements are acceptable. If, nevertheless, the memory resources are a limiting factor, the data set can also be transformed into a single cell array without prior view-dependent preprocessing. This, of course, slightly increases rendering time (See Table 5) , but still gains results significantly faster than conventional approaches of comparable image quality, while requiring just twice as much memory as the original data set.

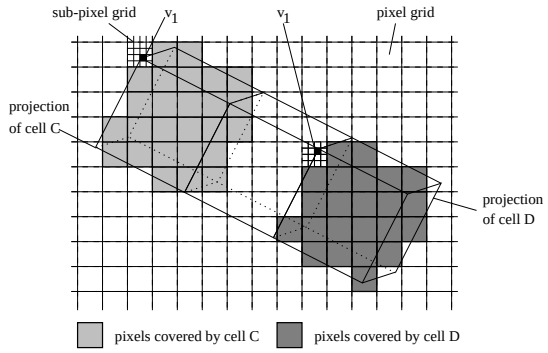


Figure 5: *Cell projection templates. As the projections of cell C and cell D (position of v_1) have different sub-pixel offsets, the shape of pixel-sets affected by the cells differs. Thus C and D require different templates for projection.*

4. Rendering

To achieve maximum rendering performance, precalculated templates are used to determine all pixels of the image which are covered by a cell's projection (Section 4.1). A fast parallel projection is used to calculate the position of a cell's projection in the image (Section 4.2). Finally, the order of traversal of the cell array and a fast heuristic estimation of the upper-bound for the maximum value along a cell-ray intersection reduce the number of more costly and accurate trilinear maximum evaluations required (Section 4.3).

4.1. Template Calculation

The main purpose of the template is to allow fast identification of the pixels affected by the projection of a cell. At each of these pixels, the cell's influence on the maximum of the ray through this pixel and thus to the pixel value has to be evaluated. A sufficiently accurate calculation of this contribution requires several steps of trilinear interpolation along the ray within the cell. To save time during cell projection, it is quite useful to pre-calculate entry and exit coordinates of the ray for each pixel of the template. Interpolation weights (u, v, $w \in [0,1]$ for trilinear interpolation) of the entry and exit points are stored for this purpose.

As only parallel projection is used, the shape and size of the projected image of all cells is identical in a continuous image space. Due to arbitrary scaling and rotation of the volume for viewing and the discrete nature of a pixelized image, images of cells differ by an individual sub-pixel displacement with respect to the pixels of the image (see Figure 5), which also leads to differing sets of pixels covered by the cell's image. To account for this shift with sufficient accuracy, the projection has to be performed with sub-pixel accuracy, allowing to place a cell's image in between image-pixel positions. The placement on a 4x4 grid within a pixel produces satisfying results.

The placement of cell images on sub-pixel positions leads to slightly different shapes of the templates for different x/y displacements and also requires the calculation of individual ray entry/exit positions for each of the templates. The resulting (4x4) array of templates can be directly accessed during rendering using the sub-pixel displacements of a cell's projection.

To optimize the rendering performance each element (=pixel) of the template stores a set of values:

- the offset of this element's pixel from the pixel containing the projection of the cell's origin (cell vertex v_1). As the image of v_1 can be located anywhere within the cell's image depending on the viewing direction, the pixel offsets may also become negative.
- the interpolation weights (u,v,w) for ray entry and exit coordinates at this element.
- the number of interpolation steps required along the ray / cell intersection
- A (du,dv,dw) vector defining a step along the ray.

Before rendering, the template is optimized to speed up access. To avoid the necessity of skipping non-covered pixels within a template, just a list of covered template entries is stored instead of a 2D array. Thus, each template is just an array of image offset / ray information elements for locations covered by a cell's projection.

4.2. Projection

As the parallel projection of a point can be performed by independently projecting it's x, y, z coordinates,

$$P\begin{pmatrix} x \\ y \\ z \end{pmatrix} = P\begin{pmatrix} x \\ 0 \\ 0 \end{pmatrix} + P\begin{pmatrix} 0 \\ y \\ 0 \end{pmatrix} + P\begin{pmatrix} 0 \\ 0 \\ z \end{pmatrix}$$

The projected positions x_{img}, y_{img} can be precalculated for the projection of all possible x, y, and z cell coordinates within the volume. This results in six arrays - one for each x_{img}, y_{img} position of each x, y, and z coordinate.

For performance reasons we use integer positions. As the summation of 3 integer values to obtain the x_{img} or y_{img} position of a cell's projection introduces an error of up to 1.5 pixels in x_{img} and y_{img}, the arrays have to contain sub-pixel coordinates to keep the error below one pixel.

In combination with precalculated templates, the projection of a cell becomes simple and efficient. `Cmax` is obtained in a way described in Section 4.3, `img_width` is the width of the image in pixels. `.pixel` is the x or y coordinate of an image pixel, `.subpix` is a sub-pixel offset.

```
(xp,yp)=projection(cell.v1);
imgpos=xp.pixel+yp.pixel*img_width;
template=subtemplate[xp.subpix, yp.subpix];
for all elements in template
{
```

```
  if(image[imgpos+
     template_element.imgoffset]<Cmax)
    evaluate cell contribution at this pixel
}
```

4.3. Evaluation of the Maximum

Compared to the other stages of MIP generation, the evaluation of maximum values within cells (trilinear interpolation) is by far the most expensive part. The reduction of the number and effort of evaluations required to generate an image is crucial for performance. Performance can be improved on one hand by using a less expensive but usually also less accurate evaluation method to approximate the maximum. On the other hand, more evaluations can be omitted if a good guess for the ray maximum can be found early. The sorted cell array allows to access and render most promising cells first. If the rendering is started with the projection of cells which have the highest cell maximum and minimum, the probability of having to evaluate successive cells projected on the same pixels is significantly reduced. As can be seen in Table 4, only about 2-4% of the cells of the original data set require the use of trilinear interpolation to evaluate their possible contribution to a MIP. The evaluation of the remaining cells is stopped either after checking C_{max} or after performing the slightly more expensive maximum estimation described below.

Values of pixels covered by the current cell which are lower than the cell maximum potentially have to be replaced by a higher value. An analytical solution for the maximum along the ray through the pixel is extremely expensive, and a few trilinear interpolation steps along the ray are also quite costly. A cheap estimation of an upper bound for the ray maximum which is more restrictive than the cell maximum C_{max} can greatly reduce the number of more costly exact evaluations of the maximum. We facilitate the following observations to present such a heuristic:

- If the maximum is located on the entry or exit point of the ray: applying two bilinear interpolations on the entry and exit faces of the cell gains the exact value for the maximum as $max(entry, exit)$.
- If for this ray the maximum is located within the cell, there must be some positive deviation from a linear evolution between the entry and exit point of the ray. If a cheap approximative guess for this nonlinearity within the cell can be found, the upper-bound of the ray can be estimated as $min(C_{max}, max(entry, exit) + deviation)$.

A fast approximative estimation of this deviation is $deviation = Max(0, Max((v_i + v_j)/2) - c)$ with v_i, v_j being data values at the vertices located at the ends of the four space diagonals of the cell and c being the trilinearly interpolated value at the center of the cell (cheap, as special case). Although this estimation is not a strict upper bound in all cases (99%), no visible impact on images of real-world data sets has been found. On average, this estimation for the ray maximum within a cell is 30% lower compared with C_{max} as an estimation. When using this estimation about 60% less of significantly more expensive trilinear evaluations have to be performed (Table 4). As only 25-30% of the trilinear evaluations lead to a change of a pixel value, a more tight upper bound estimation could gain even more performance. If the estimated upper bound for the ray is above the value of the examined pixel, several steps of trilinear interpolation within the cell are performed utilizing information stored in the templates to obtain the ray maximum.

method	*interpol.*	*cells*	*pix.set*
brute force	100%	100%	24
$+C_{max} > max$	26%	29%	24
+ignore noise	7.7%	7.8%	13
cell array	1.3%	2.5%	1.65

Table 2: *Comparison of interpolation efficiency: brute force (row 1) interpolates at every step along ray (col. 1), within every cell (col. 2). Each pixel changes 24 times (col. 3). Row 2: interpolation only if $C_{max} > max$. Row 3: also ignore low-valued cells. Row 4 shows the results for our approach.*

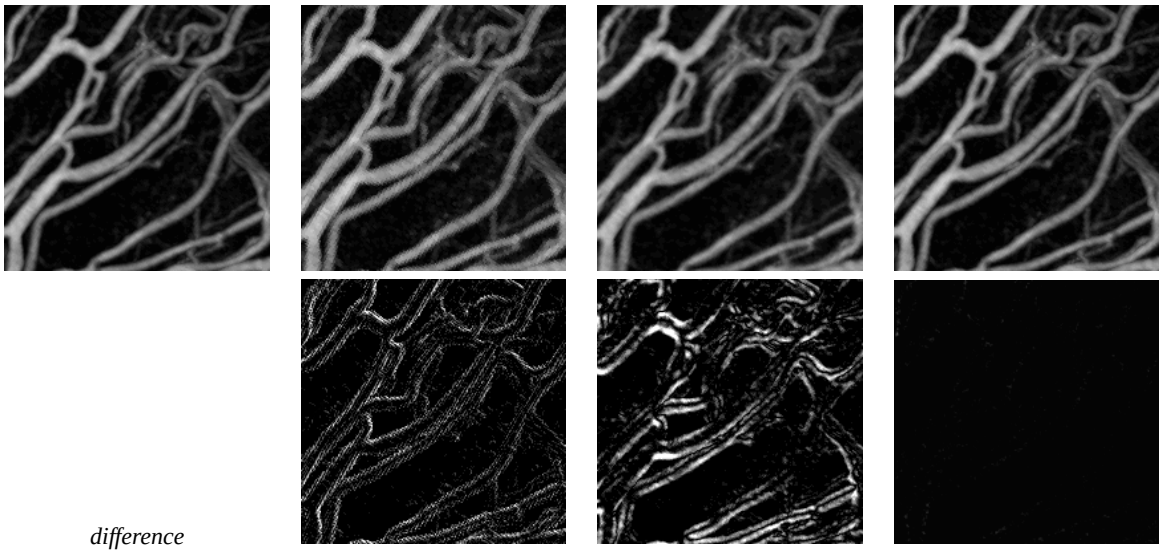
data set	*size*	*toler.*	*sweep 1*	*sweep 2*	*total*
hand	256^2x100	1%	81%	3%	84%
hand	256^2x100	2%	93%	2%	95%
kidney	256^2x69	1%	56%	5.5%	61%

Table 3: *Cell elimination results*

Projecting high valued cells first and using an additional estimation of the ray/cell maximum is very efficient, as the value of each non-background pixel of the image is set only 1.3 to 4 times compared to 10-25 times for MIP using conventional ray-casting with optimizations.

For the following pseudo-code summary of the projection of the cell array, `gray[]` stores the mapping from data-values to gray levels defined by the windowing function.

```
cells=get_cell_set(viewmatrix);
calculate_template(viewmatrix);
calculate_projection_arrays(viewmatrix);
for Cmax=highest to 0
  if(gray[Cmax]>0)
    for cell=cells.first_cell_with_Cmax
           to cells.last_cell_with_Cmax
      project(cell);
```



difference
a) trilinear interpolation *b) n. neighbor ray casting* *c) n. neighbor shear-warp* *d) cell array*

Figure 6: *Quality comparison of MIP produced using a) ray casting with trilinear interpolation, b) ray casting with nearest neighbor interpolation, c) shear-warp projection with nearest neighbor interpolation, d) sorted cells and trilinear interpolation. The difference images depict amplified difference with respect to ray casting with trilinear interpolation.*

levels of our approach	*preprocessing time*	*memory factor*	*speed-up compared to* *brute force*	*+$C_{max} < max$*	*+ignore noise*
cell array	6s	2.0	1.3-1.4		
+$C_{max} < max$	6s	2.0	6.1-7.4	3.9-5.3	3.1-4
+ignore noise	6s	2.0	20-40	14-25	11-20
+cell elimination	93.6s	3.6-12	28-43	18-27	24-22

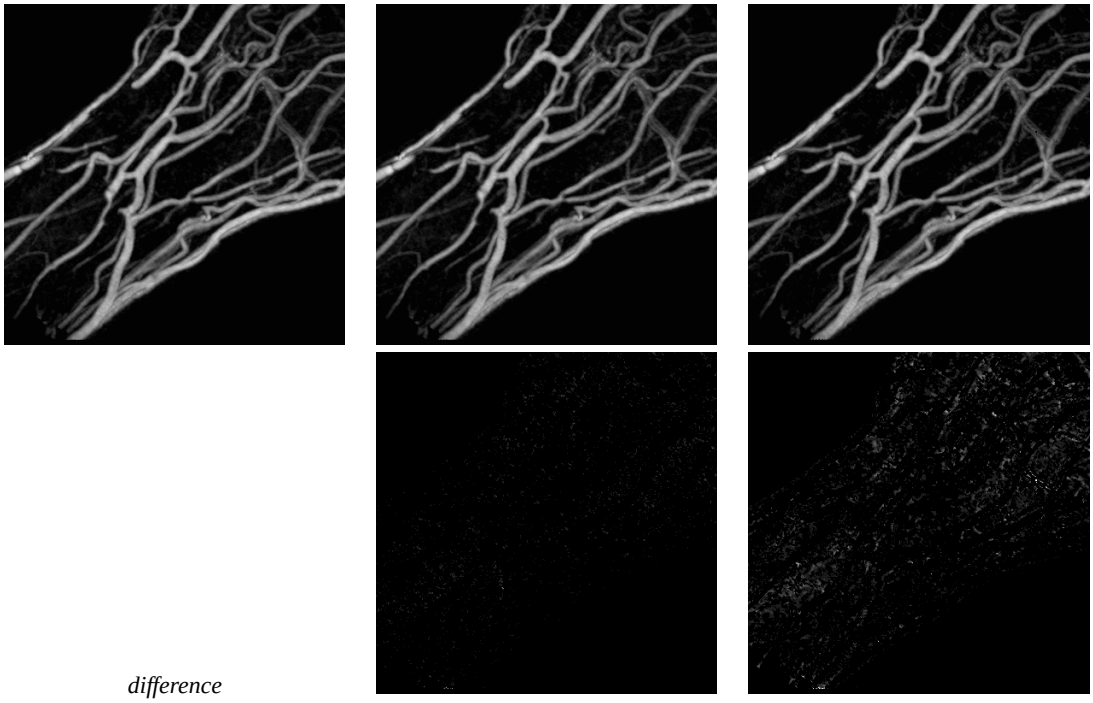
Table 1: *Comparison of ray-casting based MIP (trilinear interpolation, columns 4 and 5 with optimizations) with our approach (cell array and optimizations).*

data	*img. size*	*cell acc.*	*appr.*	*eval.*	*pixel set*
hand	465^2	2.5%	1518066	26%	1.65
hand	512^2	4.1%	2439476	27%	1.5
kidney	465^2	3.2%	1077898	29%	1.46

Table 4: *Efficiency of maximum value evaluation: Column "cell acc." gives the percentage of cells involved in the evaluation process. "Appr." is the total number of approximation operations, "eval" is the percentage of approximated ray intersections which required more exact evaluation. "Pixel set" is the number of writes to a non-background pixel.*

data set	*size*	*img size*	*no prepr.*	*prepr.*
hand	256^2x100	256^2	295ms	215ms
		512^2	615ms	450ms
kidney	256^2x69	256^2	180ms	150ms
		512^2	350ms	315ms
head	256^2x124	256^2	180ms	170ms
		512^2	380ms	365ms

Table 5: *Rendering times for different data sets and image sizes. all data sets have been preprocessed with a 1% tolerance threshold. The timings depend on the used window definition. Some of the resulting images can be seen in Figures 6d and 8a, b.*



difference tolerance 0% | *tolerance 1%* | *tolerance 2%*

Figure 7: *Results for sorted cell array MIP with different tolerance values for preprocessing (hand data set). As tolerance is increased, deviations appear mainly within areas containing low-valued noise.*

5. Results

The MIP algorithm presented in this paper has been implemented as a Java applet. The applet and further high-resolution results are available at (*http://www.cg.tuwien.ac.at/research/vis/vismed/CMIP/*). We expect a C implementation to exhibit a 30-40% better performance.

Table 1 depicts the speed-up achieved by our method compared to brute-force ray casting with trilinear interpolation (integer arithmetic) and to optimized ray-casting (no interpolation in cells with $C_{max} < max$, skip background noise windowed to black). The second row (sorted cells, evaluation only if $C_{max} \geq max$) clearly shows the advantage of projecting high-valued cells first (factor 3-7). Even more is gained by the ability to skip low-valued cells without any overhead (row 3). This results in a speed-up factor of 11-40. Finally, by eliminating cells during the preprocessing step, factors of up to 43 compared to brute force and 24 compared to optimized ray-casting can be achieved.

Rendering times as compared in Table 1 may depend on the optimization of the implementation. To obtain a less implementation dependent measure for the efficiency of the approach, we compare the number of interpolations performed, cell and image-access statistics of our algorithm to ray-casting in Table 2. Compared to even optimized ray-casting, the cell array requires five times less interpolations and changes ray-maxima and thus pixel values significantly less frequently. Due to the difference in the number of interpolations performed, we can conclude that a speed-up factor of approximately five is related to the avoidance of cell evaluations. The remaining portion of the speed-up is based on efficient volume traversal (strictly linear access to the cell array) and cell skipping.

Table 3 shows cell removal statistics for the preprocessing of the hand data set depicted in Figures 6 and 7 and the kidney data set of Figure 8a. As the second sweep eliminates just few additional cells, it can be optionally omitted to shorten the preprocessing. Increasing the tolerance for preprocessing mainly affects low-gradient areas which usually do not contain vessel data (Figure 7).

Table 4 presents statistical information on the number of cells involved in the image creation, the number of upper bound estimations for ray-cell intersections and the percentage of intersections which require a more accurate trilinear interpolation to evaluate the maximum within a cell. Rows 1 and 2 represent the hand data set for two different image sizes, Row 3 shows data for the kidney data set.

Finally, Table 5 presents rendering times for MIP using the cell array approach. The rendering times have been measured on a PIII/750 PC.

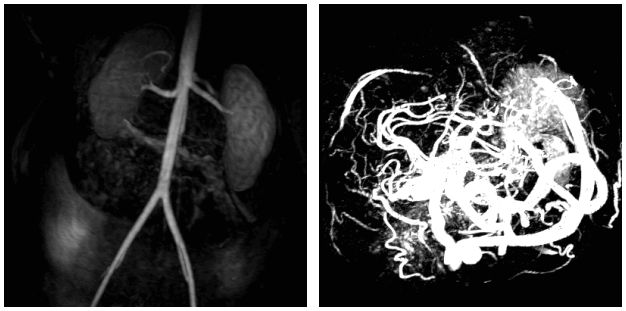


Figure 8: *a) kidney data set b) head data set*

6. Future Work

The performance of MIP methods based on sorted cell arrays can be still improved by further development in two areas: a more selective preprocessing technique, which is able to remove more cells, can reduce memory requirements and unnecessary processing of invisible cells. As the test results discussed in Section 4.3 and summarized in Table 4 show, more rigid upper bound approximation techniques could eliminate even more exact maximum evaluations. Further research should be done on these topics.

7. Conclusion

In this paper we have presented a new high-quality method for the generation of MIP images. During a preprocessing step cells which do not contribute to any MIP (regardless of the viewing direction) are identified and removed. To minimize dependencies and maximize the number of irrelevant cells, the preprocessing is performed for 12 distinct clusters of viewing directions. The positions of the remaining cells are sorted according to the cell maximum. For rendering, the cluster containing the current viewing direction is selected and projected. The cells are processed starting with cells with high values using precalculated templates to determine pixels affected by the projection. As pixels of the image are initially set to high values by the projection of high-valued cells, many unnecessary maximum evaluations which are performed by other MIP approaches can be avoided. To further decrease the number of costly maximum evaluations, a cheap and effective estimation of the upper bound for the maximum of a ray through a cell is performed first.

The method provides interactive frame-rates on high-end PCs and is well-suited for fast generation of animation loops especially to depict small details within the data which may be missed or deformed using other fast, but less accurate methods. Compared to ray-casting based high-quality MIP approaches our method avoids trilinear interpolations in a significantly more efficient way and achieves at least 20 times better performance, while providing better quality than other (shear-warp or hardware based) approaches.

8. Acknowledgements

The work presented in this publication has been funded by the FWF as part of project P-12811/INF (BandViz project) and by the $V^{is}M^{ed}$ project (`http://www.vismed.at`) which is supported by *Tiani Medgraph*, Vienna, `http://www.tiani.com`, and the FFF, Austria. The hand and kidney data sets are courtesy of Tiani Medgraph GesmbH, Vienna. The head data set is available from the United Medical and Dental Schools (UMDS) Image Processing Group in London `http://www-ipg.umds.ac.uk/archive/heads.html`

References

1. W. Cai and G. Sakas. Maximum intensity projection using splatting in sheared object space. In *Proceedings EUROGRAPHICS '98*, pages C113–C124, 1998.
2. B. Csebfalvi, A. König, and E. Gröller. Fast maximum intensity projection using binary shear-warp factorization. In *Proceedings of the 7-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media'99, WSCG'99*, pages 47–54, 1999.
3. W. Heidrich, M. McCool, and J. Stevens. Interactive maximum projection volume rendering. In *Proceedings Visualization '95*, pages 11–18, 1995.
4. J. T. Kajiya. Ray tracing volume densities. In *Proceedings of ACM SIGGRAPH'84*, pages 165–174, 1984.
5. P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorisation of the viewing transform. In *Proceedings of ACM SIGGRAPH '94*, pages 451–459, 1984.
6. W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of ACM SIGGRAPH'87*, pages 163–189, 1987.
7. L. Mroz, A. König, and E. Gröller. Real time maximum intensity projection. In *Data Visualization '99, Proceedings of the Joint EUROGRAPHICS - IEEE TCCG Symposium on Visualization.*, pages 135–144, 1999.
8. H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler. The volume pro real-time ray-casting system. In *Proceedings of ACM SIGGRAPH'99*, pages 251–260, 1999.
9. G. Sakas, M. Grimm, and A. Savopoulos. Optimized maximum intensity projection. In *Proceedings of 5th EUROGRAPHICS Workshop on Rendering Techniques*, pages 55–63, Dublin, Ireland, 1995.
10. K. J. Zuiderveld, A. H. J. Koning, and M. A. Viergever. Techniques for speeding up high-quality perspective maximum intensity projection. *Pattern Recognition Letters*, 15:507–517, 1994.