Fast Maximum Intensity Projection using Binary Shear-Warp Factorization

Balázs Csébfalvi[1], Andreas König[2], Eduard Gröller[2]

[1]Department of Control Engineering and Information Technology,
Technical University of Budapest,
Budapest, Műegyetem rkp. 11, H-1111, HUNGARY
cseb@seeger.fsz.bme.hu

[2]Institute of Computer Graphics, Vienna University of Technology,
Vienna, Karlsplatz 13/186/2, A-1040, AUSTRIA
{koenig, groeller}@cg.tuwien.ac.at

ABSTRACT

This paper presents a fast maximum intensity projection technique based on binary shear-warp factorization. The proposed method divides the density domain into a small number of intervals, and to each interval a binary code representation is assigned. In a preprocessing step, an additional volume is created which contains for each voxel the code of the interval enclosing the given voxel density. We present an appropriate data structure for storing this volume and an efficient lookup table technique which can be used to rapidly access a voxel of a certain density code. The volume is efficiently resampled along viewing rays only in voxels where the densities reside in the interval which contains the appropriate maximum value.

Keywords: Maximum Intensity Projection, Volume Rendering, Shear-Warp Factorization.

1 INTRODUCTION

Maximum Intensity Projection (MIP) is a widely accepted volume rendering technique which can be used especially for the visualization of location, shape, and topology of blood vessels. Although the shaded surfaces of different tissues cannot be rendered applying this method, practically it is much more important in diagnosis then the classical transfer function based rendering. Therefore a MIP rendering module is part of almost every commercial medical imaging system. On the other hand it can be considered as a simulated X-Ray rendering method since it approximates the density integrals along the projection rays. Therefore it can be used for general CT or MRI medical data sets as well. Although maximum intensity projection is a very useful technique for the analysis of patient data, not many efficient algorithms exist for finding the maximum intensity along a mathematical ray. In this paper a new approximate MIP method is presented, which provides interactive frame rates without using any specialized hardware.

2 PREVIOUS WORK

The classical brute force implementation of MIP resamples the density volume at a finite number of evenly located sample points along the rays and selects the maximum density sample, assuming a piecewise constant approximation of the density function. The accuracy depends on the sampling density and the applied resampling filter as well. Instead of the computationally expensive tri-linear interpolation a fast nearest neighbor resampling can be used, which provides approximately the same image quality.

In order to achieve higher accuracy, for each cell intersected by the given ray the exact entry and exit points need to be determined, and the local maximum value has to be found in the ray segment bounded by these points [14]. Considering the volume as a continuous 3D function, where the density of the intermediate points is the tri-linear interpolation of the densities of the eight closest voxels, the density function along the internal ray segment is a polynomial of third degree. The local maximum location can be calculated analytically or using heuristic approximations [14]. Exploiting that the interpolated density cannot be greater than the maximum density of the eight closest voxels this calculation has to be performed only for the promising cells, where the maximum corner density of the cell is greater than the current ray density. In spite of this, algorithms investigating the local maximum locations inside a cell are rather time demanding, therefore they cannot be applied in interactive applications.

In the last two decades several volume rendering techniques have been published aiming at the accelerated evaluation of the well known light transport equation [2][3][4][6][7][8][11][12][13]. Most of these methods use hierarchical data structures to speed up the ray traversal exploiting the coherence of the data set [2][5][9]. The min-max versions of these data structures like octrees, K-d trees, or pyramids can be used for accelerated maximum intensity projection as well. Among the direct volume rendering acceleration techniques, the greatest time savings have been made with Lacroute's shear-warp factorization algorithm, based on the run-length encoding of transparent regions [4]. Since this method was proposed for the fast alpha-blending evaluation, it cannot be used for MIP without modification. The new method presented in this paper also uses the shear-warp projection approach, but the preprocessing is performed in a completely different manner optimized for maximum intensity projection.

3 THE ALGORITHM

The main problem in maximum intensity projection is that we have to investigate several volume samples along each viewing ray and only one sample per ray contributes to the final image. In order to avoid the unnecessary resampling our method decomposes the density domain into a finite number of intervals. At first, for each ray the interval is determined which contains the maximum density, then the volume is resampled at those voxel locations where the densities reside in the given interval.

For the sake of clarity, we consider the volume to be a piecewise constant 3D density function and later we discuss how to extend the method to use a more exact bilinear interpolation for resampling. To each voxel a binary code is assigned representing an interval which contains the density of the given voxel. The density domain is divided into n intervals I_0, I_1, ... I_{n-1} sorted in ascendanting order by their borders with increasing index. To each interval the binary format of the corresponding index is assigned as a unique code. In a preprocessing step an additional volume is created which stores the density interval codes for each voxel.

In the practice of evaluation of medical data sets usually a *windowing* function is used to map the relevant part of the density domain onto intensities (Figure 1)(for the sake of simplicity, in the further discussion we will use the "density" term for the data values of the input array, although it is not completely correct for MRI data sets). This function is defined by two parameters, a lower (L) and a higher (H) thresholds. In order to handle the irrelevant part efficiently the intervals I_0 and I_{n-1} are defined as [0,L) and [H,M] respectively, where M is the maximum density value. The domain [L,H) can be subdivided using a uniform or a balanced quantization based on the histogram of the data set.

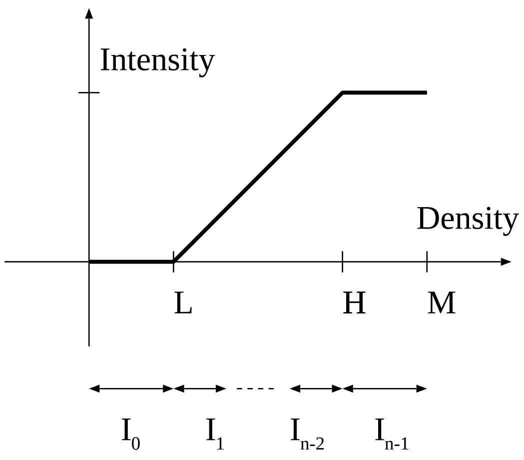

Figure 1: Mapping the densities onto intensities.

Without loss of the generality, assume that the number of intervals is four ($n = 4$). In this case the intervals can be encoded on two bits, where the code 00 represents the lowest and 11 the highest interval. The binary volume containing the corresponding codes for each voxel can be stored in a 3D integer array *mask*, where an integer element contains continuously the codes of voxels along the rows parallel to the *z-axis*. Having a density volume of size $X \times Y \times Z$ this integer array is defined as *int mask*[*depth*][*height*][*width*], where *depth = (2 * Z + 31) div 32*, *height* = Y, *width* = X and the size of an integer is supposed to be 32 bits. The following two subsections describe how to use such a binary volume for fast global and local maximum intensity projection assuming that the viewing direction is the *z-axis*. Later we discuss how to extend this method to arbitrary viewing directions using binary shear-warp factorization.

3.1 Maximum Intensity Projection

In order to project the maximum intensities onto the x, y plane, rays are cast into the z direction from each (x_i, y_j) grid point. Since in this special case, the sample points are located at voxel positions, the integer elements of the array *mask* store the density interval codes of the samples along the rays parallel to the *z-axis*. For each ray, first that interval is determined which contains the maximum density in order to avoid the resampling in the lower density intervals. The following routine compares bit patterns to find the code of the highest density interval, checking 16 samples in each step of the loop.

```
int MaxInterval(int i, int j) {
 int index = 0;
 for(int k = 0; k < depth; k++) {
  int segment = mask[k][j][i];
  if(segment) {
   int I01orI11 =
    segment & 0x55555555;
   if(I01orI11) {
    if(I01orI11 & (segment >> 1))
     return 3;
    else if(index < 1) index = 1;
   }
   else if(index < 2) index = 2;
  }
 } return index;
}
```

The variable *I01orI11* is greater than zero if the integer *segment* contains at least one 01 or 11 pattern. In this case, if the bitwise and operation of *I01orI11* and the *segment* shifted right is greater than one then there is at least one 11 found pattern otherwise the highest density interval code is 01. If the value of *I01orI11* is zero then the highest density code is 00 or 10 depending on whether the value of *segment* is zero or not. Having four or more intervals a lookup table can be used in order to find the bit pattern inside an integer representing the highest density interval. The lookup table stores the index of the corresponding interval for each byte combination and the bytes in an integer are checked sequentially. After having the appropriate interval code only those samples have to be investigated, which have the same density code. For example, if the return value of *MaxInterval* is 1 the bit pattern 01 has to be searched for in the integer array *mask*. According to the offset of the found pattern the location of the corresponding voxel is determined, and the exact intensity value is compared with the current ray intensity.

```
int MIP(int i, int j) {
 int max = 0;
 int pat = MaxInterval(i,j);
 if(!pat) return 0;
 for(int k = 0; k < depth; k++) {
  int seg = mask[k][j][i], pos;
  while((pos = Offset(seg,pat)) < 32) {
   int density =
   volume[k * 16 + pos / 2][j][i];
   if(density > max) max = density;
   seg |= ~(0xC0000000 >> pos);
  }
 } return max;
}
```

The routine *Offset* returns with the offset of the found bit pattern passed as a second argument. Having four lookup tables of 256 byte size storing the position of the first pattern inside a byte for each interval code, this operation can be performed very fast (Figure 2).

```
int Offset(int segment, int pattern) {
 int pos, *lut = SelectLUT(pattern);
 pos = lut[segment >> 24];
 if(pos < 8) return pos;
 pos = lut[(segment << 8) >> 24];
 if(pos < 8) return pos + 8;
 pos = lut[(segment << 16) >> 24];
 if(pos < 8) return pos + 16;
 pos = lut[segment & 0x000000FF];
 if(pos < 8) return pos + 24;
 return 32;
}
```

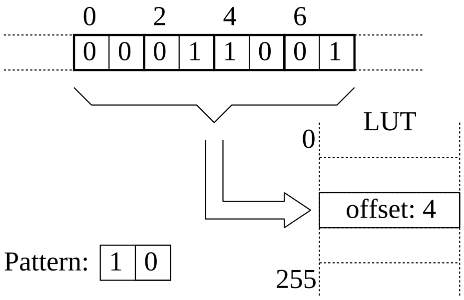

Figure 2: An example for a LUT entry .

Assume that, if the given byte does not contain the searched pattern then the lookup table stores 8 for this bit combination. On the other hand, if the whole segment represented by an integer does not contain the given pattern then the return value of the routine *Offset* is 32. The address of the applied lookup table is returned by the routine *SelectLUT* depending on the bit pattern required to be searched. Note that, the processed samples are deleted from the variable *seg*, thus this lookup table technique can be used again to determine the location of the next sample.

3.2 Local Maximum Intensity Projection

For local maximum intensity projection (LMIP) which is an extended version of

MIP a similar lookup table technique can be used in order to reduce the number of samples to be taken. LMIP selects the first local maximum along a ray which is greater than a pre-defined threshold [15]. With an appropriate parameter setting LMIP can provide similar shading effects as the transfer function based volume rendering and more precise geometric information like the depth and occlusion of vessels can be obtained.

In this case, just one bit is assigned to each voxel indicating whether the density is under or above the pre-defined threshold, thus the problem of finding the first density sample along a ray which is higher than the threshold is reduced to the problem of finding the position of the first non-zero bit inside an integer. Having a lookup table storing this position for each byte combination we have to check the bytes of the given integer sequentially, and the lookup table is addressed by the first non-zero byte. Taken into account the offset of the first non-zero byte and the read value the z position of the first sample can be calculated easily. The further samples are investigated sequentially until the first local maximum is found. According to the definition of LMIP, for those rays which do not intersect any voxels of densities higher than the threshold the global maximum has to be taken. For these rays the data structure defined in the previous section can be used. Although for LMIP we cannot apply a more precise density encoding, due to the early ray termination it is usually faster than the traditional MIP.

3.3 Shear-Warp Factorization

The previous method works only for a special case but it can be extended to viewing directions, where the principal component is the coordinate z using binary shear-warp factorization. This transformation effectively moves the bits of the integer array *mask* perpendicularly to the viewing direction.

In an interactive volume rendering application the volume is required to be rotated continuously by small difference angles. Adding the temporal dimension of animation enables the user to perceive the topology of the surfaces much better then in a static image. If the difference angle is small enough then there is no slice in the binary volume which has to be shifted by more than one bit. In this case, a single shear operation can be performed very efficiently, since just bitwise operations need to be executed on neighbor integers. That is the reason why our method shears the binary mask incrementally, applying a technique proposed by Cohen-Or and Fleishman [1]. They used their so called incremental alignment algorithm in order to reduce the communication overhead in a large multi-processor architecture supporting shearing of volumes.

As some bits can be shifted out of the array, it has to be extended by $X/2$ bits along the *x-axis* and by $Y/2$ bits along the *y-axis* in both directions as it is depicted in Figure 3.

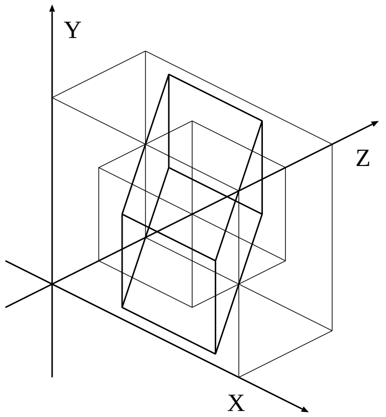

Figure 3: Extension of the binary mask.

This extended array is defined as: *int mask*[*depth*][*height*][*width*], where $depth = (Z + 31)\ div\ 32$, $height = Y * 2$ and $width = X * 2$. The routine *ShearLeft* demonstrates, how to execute one shear step in the left direction.

```
void ShearLeft() {
  int i, j, k;
  for(k = 0; k < depth/2; k++) {
    for(j = 0; j < height; j++) {
      for(i = 0; i < width-1; i++)
        mask[k][j][i] =
        mask[k][j][i] & shift_x[k] |
        mask[k][j][i+1] & ~shift_x[k];
      mask[k][j][width-1] &= ~shift_x[k];
    }
  for(k = depth/2; k < depth; k++) {
    for(j = 0; j < height; j++) {
      for(i = width-1; i > 0; i--)
        mask[k][j][i] =
        mask[k][j][i] & shift_x[k] |
        mask[k][j][i-1] & ~shift_x[k];
      mask[k][j][0] &= ~shift_x[k];
    }
}
```

For the sake of clarity this routine is not optimized, but it can be improved introducing local pointer variables in order to avoid unnecessary array addressing, and on the other hand only those parts of the extended mask need to be sheared which contain the non-zero bits representing the non-zero intensity voxels. The integer array *shift_x* is defined as: *int shift_x*[*depth*] and it stores a binary vector of size Z indicating those z positions where the corresponding slices have to be shifted in the given shearing phase. An additional array denoted by *shift_y* is used for shearing in the y direction. In order to determine the offsets of the slices in different shearing phases, another two arrays are introduced for storing the discretized translations along the *x-axis* and the *y-axis.* They are defined as: *int trans_x*[*Z*] and *int trans_x*[*Z*] respectively. These translation arrays contain the x and y coordinates of a 3D discrete line for each z depth.

Initially, this line corresponds to the *z-axis*, thus the translation arrays contain zeros. In order to compute the discrete translation vectors a symmetric DDA line drawing like Bresenham's can be used [10]. Before executing a binary shear operation the *shift* vectors are evaluated in advance according to the rotation direction. For example, when a counterclockwise rotation around the *y-axis* is needed, *trans_x*[*0*] is incremented and *trans_x*[*Z - 1*] is decremented by one and the intermediate values are computed using a symmetric DDA line drawing connecting the new end points. Since the difference between the old and the new translations cannot be greater than one the new *shift* arrays are computed as a binary difference vector of the new and the old translation vectors. In order to avoid the cutting of the binary codes stored in the integer array *mask* the bits of the two bit long segments in the array *shift* have to be set to the same value. Since in the sheared mask a ray emitted from an (i, j) grid point on the x, y plane is represented by the integers *mask*[*k*][*j*][*i*], where *k = 0, 1, ..., depth - 1* the same MIP and LMIP techniques can be used which have been presented in the previous subsections. Because of the shear-warp projection an additional relatively cheap 2D warp operation is performed on the intermediate image producing the final image which is a parallel projection of the volume.

4 EXTENSIONS

The presented fast rotation technique can be extended to arbitrary viewing directions since a binary volume can be created for each principal direction. The perspective projection is also supported by the proposed data structure applying an appropriate scaling of the slices depending on the z depth.

Since the rays are approximated with 3D discrete lines the algorithm does not generate exact maximum intensity projections, but it can be further improved using bilinear interpolation for computing the density samples. In this case the binary density codes have to be assigned to rectangular cells on the planes perpendicular to the principal component of the viewing direction rather then to voxels, where the code of a cell is determined by the lowest density corner voxel. In the resampling process, whenever the searched bit pattern is found in the array *mask* its location is mapped onto the corresponding cell location and the sample density is computed from the four corner densities using bilinear interpolation.

Using four bit codes the shear operation is slower but since the binary volume contains more exact information about the voxel densities much less samples have to be taken in the projection step. Ignoring the complete resampling process the highest density indices along the rays can be displayed directly yielding an image with 16 gray levels, thus the method can be used as a fast MIP pre-viewer.

5 IMPLEMENTATION

The presented algorithm has been implemented in C++ and has been tested on an SGI Indy workstation. Table 1 summarizes the average shearing, MIP, and LMIP running times per frame. We used two data sets, resampling the same CT scan in different resolutions, where the size of the file *smallhead* is $128 \times 128 \times 113$, while the file *bighead* has a $256 \times 256 \times 225$ resolution.

data set	smallhead	bighead
shearing time	0.011 sec	0.104 sec
LMIP time	0.077 sec	0.432 sec
MIP time	0.193 sec	1.812 sec

Table 1: Running time measurements.

Since the binary shear transformation is relatively cheap computationally the effective speed of the rotation can be increased performing a couple of incremental shears between the frame generations. The average frame rate of a smooth rotation is approximately 10 Hz for the small resolution data set and 2 Hz for the high resolution volume using LMIP rendering.

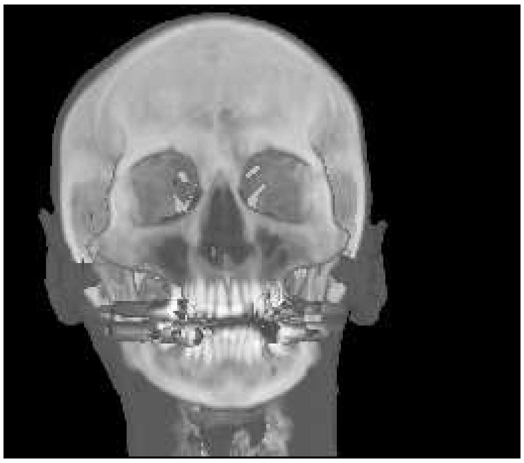

Figure 4: LMIP of a CT scan of a human head.

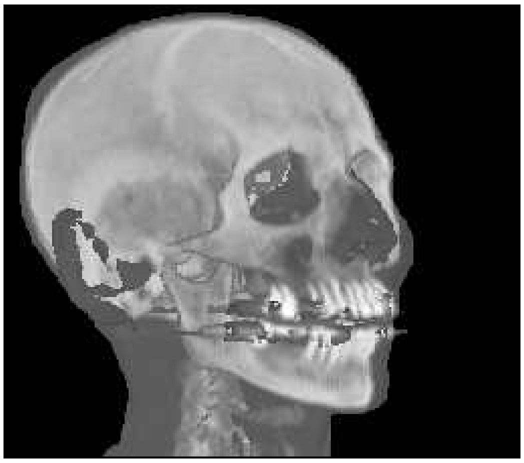

Figure 5: LMIP of the rotated data set.

Figure 4 shows the high resolution data set rendered using LMIP and Figure 5 shows the projection of the rotated volume. Due to an appropriate filtering, point like noise artifacts in the acquired data were not considered as local maxima.

6 CONCLUSION

In this paper a new fast maximum intensity projection technique has been presented, which does not require any specialized hardware. Due to the density encoding scheme and the applied lookup table technique the number of density samples to be taken is significantly reduced speeding up the maximum intensity projection. The proposed method supports the local maximum intensity projection as well, where a depth dependent filter (like a median filter) can be used for noisy data sets.

ACKNOWLEDGEMENTS

This work has been supported by the National Scientific Research Fund (OTKA ref.No.: F 015884) and the Austrian-Hungarian Action Fund (ref.No.: 29ö4 and 32öu9) and it is part of the V^{is}M^{ed} research project (http://www.vismed.at) supported by FFF (Forschungsförderungsfond für die gewerbliche Wirtschaft) and Tiani Medgraph. The CT scan was obtained from the Chapel Hill Volume Rendering Test Dataset. The data was taken on the General Electric CT scanner and provided courtesy of North Carolina Memorial Hospital.

REFERENCES

[1] Daniel Cohen-Or and Shachar Fleishman. An incremental alignment algorithm for parallel volume rendering. *Computer Graphics Forum (EUROGRAPHICS '95 Proceedings)*, pages 123–133, 1995.

[2] John Denskin and Pat Hanrahan. Fast algorithms for volume ray tracing. *Workshop on Volume Visualization*, pages 91–98, 1992.

[3] R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics (SIGGRAPH '88 Proceedings)*, 22:65–74, 1988.

[4] Philippe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Computer Graphics (SIGGRAPH '94 Proceedings)*, pages 451–457, 1994.

[5] David Laur and Pat Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics (SIGGRAPH '91 Proceedings)*, pages 285–288, 1991.

[6] Marc Levoy. Display of surfaces from ct data. *IEEE Computer Graphics and Application*, 8:29–37, 1988.

[7] Marc Levoy. Efficient ray tracing of volume data. *ATG*, 9(3):245–261, 1990.

[8] Derek R. Ney, Elliot K. Fishman, Donna Magid, and Marc Levoy. Computed tomography data: Principles and techniques. *IEEE Computer Graphics and Application*, 8, 1988.

[9] K.R. Subramanian and Donald S. Fussell. Applying space subdivision techniques to volume rendering. *IEEE Visualization '90*, pages 150–159, 1990.

[10] L. Szirmay-Kalos (editor). *Theory of Three Dimensional Computer Graphics.* Akadémia Kiadó, Budapest, 1995.

[11] Jason Freund and Kenneth Sloan. Accelerated volume rendering using homogeneous region encoding. *IEEE Visualization '97*, pages 191–196, 1997.

[12] D. Cohen and Z. Shefer. Proximity clouds - an acceleration technique for 3D grid traversal. *TR FC93-01, Ben Gurion University, Israel*, 1993.

[13] K. Zuiderveld, A. Koning, Viergever and A. Max. Acceleration of ray casting using 3D distance transformation. *Visualization in Biomedical Computing*, pages 324–335, 1992.

[14] G. Sakas, M. Grimm, and A. Savopoulos. Optimized Maximum Intensity Projection (MIP). *Workshop on Rendering Techniques*, pages 51–63, 1995.

[15] Y. Sato, N. Shiraga, S. Nakajima, S. Tamura, and R. Kikinis. LMIP: Local Maximum Intensity Projection. *Journal of Computer Assisted Tomography*, Vol. 22, No. 6, 1998.