

Permutation Warping for Data Parallel Volume Rendering

Craig M. Wittenbrink

Dept. of Electrical Engineering,
University of Washington
Seattle, WA 98195
craig@shasta.ee.washington.edu

Arun K. Somani

Dept. of Computer Science and
Engineering
Dept. of Electrical Engineering,
University of Washington
Seattle, WA 98195
somani@ee.washington.edu

Abstract

Volume rendering algorithms visualize sampled three dimensional data. A variety of applications create sampled data, including medical imaging, simulations, animation, and remote sensing. Researchers have sought to speed up volume rendering because of the high run time and wide application. Our algorithm uses *permutation warping* to achieve linear speedup on data parallel machines. This new algorithm calculates higher quality images than previous distributed approaches, and also provides more view angle freedom. We present permutation warping results on the SIMD MasPar MP-1. The efficiency results from nonconflicting communication. The communication remains efficient with arbitrary view directions, larger data sets, larger parallel machines, and high order filters. We show constant run time versus view angle, tunable filter quality, and efficient memory implementation.

1 Introduction

Volume rendering [4] is memory and compute bound. Researchers have used parallelism to speedup transparency volume rendering. The goal in parallelism is linear speedup, and no storage overhead. Linear speedup is parallel run time t_p on P processors that is a fraction of the best sequential run time t_s , or $t_p = O(t_s/P)$. Storage efficiency is $O(S)$ storage with S sample points, equivalent to the sequential algorithm's storage.

Previous parallel volume rendering algorithms have restricted platforms [7], data set sizes [17][6], filter quality [9][12][2][10][3], view angle freedom, or correctness [13]. Users shouldn't have to sacrifice functionality to achieve higher performance with parallel computers. We have developed a parallel algorithm that uses unrestricted viewpoints and filters, with efficient storage, linear run-time speedup, and problem and generation scalability. Unrestricted viewpoints are achieved with provable one-to-one communication. For this reason we call our algorithm *permutation warping for parallel volume rendering*. This paper extends work on data parallel rotation [9] and our work on parallel image warping [15]. We have developed new decompositions for our permutation assignment, quantified the error of competing multipass shears, provided a new virtualization technique that keeps run time constant across view angle, and implemented the algorithm on the MasPar MP-1. The permutation warp provides processor and problem size scalability with linear speedup.

2 Permutation Warping

We define voxels to be point samples. Assume that there is a processor available for every sample point and define processors as $\pi[i, j, k]$, where i, j, k are integers. Assign processors to sample points $p(x, y, z)$, where x, y, z are reals. This re-

quires $P = S$ processors. Also give processors a screen space assignment of π' . We will discuss data parallel virtualization in Section 3. Our algorithm consists of the following three steps:

In **Step 1**, processors classify and shade reading neighboring data as necessary.

In **Step 2**, each processor resamples the opacities, α_p , and intensities, I_{Sp} , to be aligned with the view rays. If done in a straight forward fashion this would require many rounds of communication, but we have developed a permutation warp that requires only one communication. We resample in the object space near where the points lie, and then send the resampled data to its screen space position. For example, permutation warping uses a single one-to-one communication versus eight for a trilinear filter. The challenge to doing this is using a rule, M , to calculate processor assignments for the viewing transform [15].

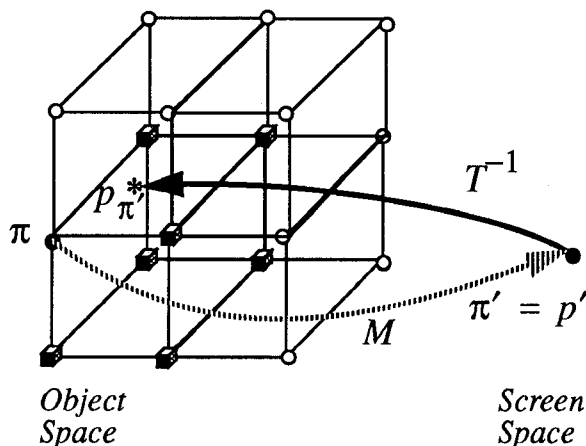


Fig. 1 Transforms and Communications in Permutation Warping for a Single Voxel

Fig. 1 illustrates the transforms calculated by a single processor. The object space and screen space are separated, the object space on the left and the screen space on the right. A processor π does permutation warping by:

- 2.1) Calculating processor assignments $\pi' = M(\pi)$; the logical connection is shown by the dotted line in Fig. 1.
- 2.2) Calculating reconstruction point $p_{\pi'} = T^{-1}(p'_{\pi'})$; the inverse transform is shown by a solid line and the point is shown as an asterisk (*).
- 2.3) Performing resampling of α_p and I_{Sp} , reading the values of I_{Sp} and α_p of its neighboring processors. The number of neighbors used determines the filter order.
- 2.4) Sending resampled values to screen processors π' .

In **Step 3**, a parallel product evaluation combines resampled intensities and opacities. Binary tree combining computes products for any associative (not necessarily commutative) operator \otimes , $I_1 \otimes I_2 \otimes \dots \otimes I_w$ [5]. Compositing (I_i over I_j) is associative. Numerical integration is also associative. In the next section we further discuss the mapping M and our generalizations from [15].

2.1 Processor Assignment by Permutation Warp

Paeth [8], Tanaka et al. [11], and Schroeder et al. [9] have used pure shear matrix decomposition of rotation to create efficient resampling algorithms. The technique is a refinement of multipass filtering where the transform is restricted to rotation. By not actually resampling data, and using the shears only to calculate the processor assignment, we use a better filter, and are still efficient. While Schroeder and Salem used a one to one assignment [9] to calculate multipass resampling, we are interested in calculating a direct one pass resampling. A permutation warp M calculates $[i', j', k']$ given $[i, j, k]$ and a transform T . To understand why we go through the extra work of calculating M , Fig. 3 shows communications taking place in parallel for a volume rotation. There are no conflicts. Each white line connects only two processors shown by the parallel nature of all of the lines. The object space processor bounding box is green, and the forward T warped version is also given as green in the screen space. The screen space processor bounding box is red in both spaces. Of course, processors are both object space processors π and screen space processors π' with $\pi[i, j, k] = \pi'[i, j, k]$. This is shown by those processors who interpolate for themselves, the blue processors in the interior where communications arcs are not drawn.

We have further qualified the transforms T permutation warping can be used for. They are the equiareal transforms defined by $\det(T) = \pm 1$ (determinant). Here we develop a solution for any 3D transform of this type.

The processor assignment is calculated by the transform $\pi' = M(\pi)$. This permutation transforms points p_π , whose coordinates are a tuple of integers, to another point, $p_{\pi'}$, whose coordinates are also integers. An integer coordinate field is mapped to another integer coordinate field, and the point $p_{\pi'}$, when inverted by T^{-1} to $p_{\pi'} = T^{-1}(p_{\pi'})$ is within π 's neighborhood. Obviously M and T are closely related. The distances satisfy $|(p_{\pi'})_x - p_\pi| < 1$, $|(p_{\pi'})_y - p_\pi| < 1$, and $|(p_{\pi'})_z - p_\pi| < 1$, a working definition of neighborhood. Our proof for two-dimensional rotation decomposition, M , is found in [15], and our code has built in neighborhood check that has not yet been violated. Proof for 3D can be done similar to 2D by brute force calculation of $p_1 = T^{-1}Mp_2$ and comparing p_1 and p_2 .

$M = M_1M_2\dots M_n$ is a concatenation of pure shear, translation, and round operators. Rounding is used to snap real values to integers. Shear are non angle preserving affine transforms. A pure shear is nonscaling and preserves distance. Any affine transform, T , with $\det(T) = 1$ is nonscaling, or equiaffine. This includes shears, rotations, and translations, that are all isometries. By allowing $\det(T) = \pm 1$, reflections can also be calculated for equiareal transforms. An isometry is a one-to-one and onto transform that preserves distances.

The general solution to a three dimensional equiareal transform is calculated by solving a system of ten equations with nine unknowns (both (EQ 1) and (EQ 2)),

$$\det(A) = \pm 1 \quad (\text{EQ 1})$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & b_{12} & b_{13} \\ 0 & 1 & b_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ c_{21} & 1 & 0 \\ c_{31} & c_{32} & 1 \end{bmatrix} \begin{bmatrix} 1 & d_{12} & d_{13} \\ 0 & 1 & d_{23} \\ 0 & 0 & 1 \end{bmatrix}. \quad (\text{EQ 2})$$

The system appears to be over constrained, but can in fact be solved. The symbolic solution from Mathematica(TM) is,

$$\begin{aligned} c_{31} &= a_{31}, & c_{32} &= \frac{a_{31} - a_{22}a_{31} + a_{21}a_{32}}{c_{21}}, \\ d_{12} &= \frac{a_{22} - 1}{c_{21}} + \frac{a_{32}}{a_{31}} - \frac{a_{21}a_{32}}{c_{21}a_{31}}, & d_{23} &= \frac{c_{21} + a_{22}a_{31} - a_{21}a_{33}}{a_{22}a_{31} - a_{21}a_{32}}, \\ d_{13} &= \frac{a_{23}}{c_{21}} - \frac{a_{23}a_{31} + a_{21}a_{33} - c_{21}}{c_{21}(a_{22}a_{31} - a_{21}a_{32})} + \frac{a_{33}}{a_{31}} - \frac{a_{21}a_{33}}{c_{21}a_{31}}, \\ b_{23} &= \frac{a_{21} - c_{21}}{a_{31}}, & b_{13} &= \frac{a_{11}}{a_{31}} - \frac{c_{21}a_{11}a_{32} + c_{21}a_{31}a_{12} - a_{31}}{a_{31}(a_{22}a_{31} - a_{21}a_{32})}, \\ b_{12} &= -\frac{1}{c_{21}} + \frac{a_{31} + c_{21}(a_{12}a_{31} - a_{11}a_{32})}{c_{21}(a_{22}a_{31} - a_{21}a_{32})}. \end{aligned} \quad (\text{EQ 3})$$

The solution above allows direct solution for a three pass non-scaling transform (EQ 2) (or six passes of single coordinate shears versus the eight passes from [9]), which we use in our permutation calculation. It could also be used for a multipass warping provided the data could move in two directions operating on scanframes.

3 Data Parallel Virtualization

To apply permutation warping without a processor for each sample point requires emulating v virtual processors on $P < v$ physical processors. We have found that a data parallel approach uses permutation warping very efficiently.

To virtualize we make an assignment of P processors to the S voxels. Object space voxel points are assigned to processor id's by an address tiling. Address tiling in three dimensions is an extension of two dimensional tiling techniques such as our cache tiling [14]. Wittenbrink in [16] provides more detail on slice and dice virtualization. A slice, row major addressed volume coordinate is transformed to a sliced and diced coordinate by $\langle t | k | r | i | s | j \rangle \rightarrow \langle t | r | s | k | i | j \rangle$. Such virtualization is amenable to a wide variety of architectures such as mesh [1], hypercube, and multistage interconnection networks. Each dimension gets approximately $P^{1/3}$ cuts. The algorithm is the same as that in Section 2, except now processors calculate M for every point. The screen space samples being calculated are unique, but processors may receive more than one message because of virtualization. The density of messages across the network is the same if the slice and dice virtualization is used and communication remains efficient.

4 Time/Quality Trade-offs

Multipass shears, [8][9][11][12], and direct warping, Section 2 [15], are not equivalent. Because each resampling discards the prior data, a shear filtering approach has more resolution error and interpolation error than a comparable direct filter. After a shear, all that is stored is the new samples. We used two test objects to calculate the reconstruction error: a cube of intensity 65535 and a sphere whose intensities are zero at the edge and 65535 at the center. 16 bit intensities were used. The volumes were 128x128x128 voxels with the sphere and cube centered and of diameter/width equal to 64. A Sun Sparc 2 was used to calculate the comparison to ease implementation of the shearing ap-

proach. The errors were calculated by differencing each sample point for an altered viewpoint with the analytically defined cube or sphere. Absolute errors were summed on each ray. We compare two direct filters: a zero order hold (zoh) and a first order hold (trilinear), and a multipass filter using linear interpolation.

The empirical study shows that the cube's mean error per ray is 3.45% zoh, 4.58% multipass, and 3.46% trilinear. The maximum error encountered on any ray rendering the cube is 9.37% zoh, 15.57% multipass, and 12.3% trilinear. For the sphere mean errors were 1.15% zoh, 0.22% multipass, and 0.07% trilinear, and max errors were 11.98% zoh, 3.04% multipass, and 1% trilinear. The trilinear is clearly better than shearing, but the zero order hold is the same as the trilinear for the cube and worse than trilinear and shearing for the sphere. This results from the frequency content of the volumes. The cube is a step function and has infinite frequencies. The zero order hold maintains the resolution very well. The multipass approach has repeated aliasing steps which degrades the reconstruction. Wittenbrink in [16] provides more details.

Fig. 4, Fig. 5, and Fig. 6 show MR angiography rendering to illustrate filter differences. Fig. 4 shows the noise inherent in the MR angiography data. Fig. 5 shows the 256x256x32 data rendered at 512x512 zooming in on the bifurcation of Fig. 4. Fast traversal is possible with the zoh of Fig. 5, and a more accurate trilinear filter is used to render Fig. 6. The filter difference on these medical image is readily apparent.

Performance measurements were taken on the MasPar MP-1 [1]. The MasPar used for the performance study was a 16384 SIMD processor MP-1 whose peak performance is 26,000 MIPS (32 bit integer) and 1,200 MFLOPS (32 bit floating point). The architecture supports frame buffers through VME frame grabbers, HIPPI connection, or through MasPar's frame buffer (not available yet). Image display in the current implementation is done on the X host. The processors are interconnected through a mesh with 23,000 Mbytes/sec peak bandwidth and a multistage crossbar router with 1,300 Mbytes/sec peak bandwidth. The array controller provides a software accessible hardware timer that accurately captures the elapsed run time.

Our implementation in MPL, a C like parallel language, uses the slice and dice virtualization discussed. The neighboring processors do not need to be accessed in the resampling step because of a one voxel overlap of volume storage on each processor. The overlap allows a random access to replace a costly case decision in the SIMD language. The storage overlapping does not reduce the size of volumes that can be processed in practice. We take advantage of the MasPar instruction ScanMax. Once each processor composites its subcube, ScanMax composites across z in segments to complete each parallel product in one instruction. The over operator can be done similarly using the Scan operator to create the proper transparency at each processor, and then doing a parallel addition.

Measurements given are the average of multiple runs at each angle. Fig. 2 shows the run times to render a 128x128x128 byte volume to a 128x128 image versus resampling angle. The zero order hold is most efficiently calculated without using permutation warping, as we showed in [15]. See TABLE 1. The rotation only times are given in Fig. 2 and TABLE 3 also showing how the resampling for rotation takes the majority of the time. The many lines for each filter show rotation about x, rotation about y, rotation about z, and rotation about x, y, and z. Each time represents rendering from the original data.

TABLE 2 gives the mean run time across all angles. Note that the performance is tightly bounded and predictable. The effectiveness of direct warps lies in the performance filter tunabil-

ity. The zero order hold takes from 73% to 146% less time than the first order hold, and can be used for interactive performance in viewing the larger volumes. The trilinear interpolation, or first order hold, has comparable performance to the multipass warps but is more accurate. Fig. 5 and Fig. 6 show the filter quality tuning for the foh and zoh.

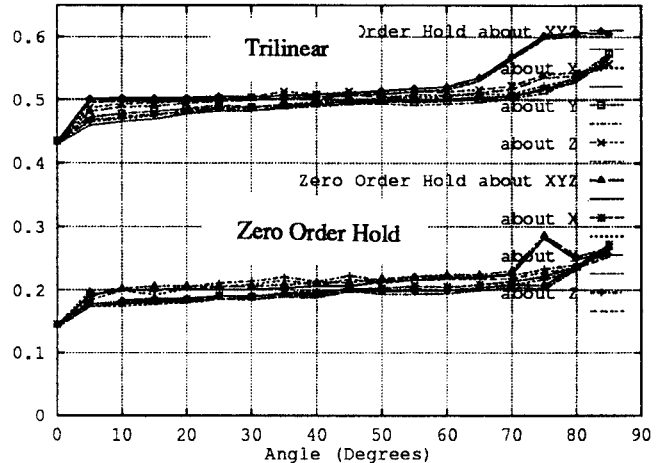


Fig. 2 Nearly Constant Run Time Versus Angle

Communication congestion is low for the data parallel permutation warp. There is no congestion if a processor is available for every sample point. Using the rotation speed of 0, 0, 0 degrees in TABLE 1 as zero congestion the congestion is 19% to 29% of the run time for permutation warping, foh. The congestion is 40% to 43% for the backwards algorithm, or zoh. The router start-up penalty and/or the rule overhead account for the rest of the difference.

The closest comparisons are to [12][9][10] who use similar voxel sizes and architectures. Comparison of resampling times shows that direct filters cost more [12] but the direct filter is superior to the shear locally then send approach [9] with up to a five times speedup depending on the machine. The forward wavefront approach [2][10] trades view angle freedom for high performance and a slight speedup over our work. We have through permutation warping provided improved quality and view angle freedom for data parallel machines.

5 Conclusions and Future Work

We presented an EREW PRAM algorithm for volume rendering, and demonstrated its efficiency on a parallel machine. Our general reconstruction filter approach provides for time/quality trade-offs not possible in previous data parallel approaches making data parallel implementations more useful for volume rendering.

The data parallel version can be ported to nearly all massively parallel general purpose computers. This fact, and the ability to change combining rules, shading, or reconstruction filters, shows that permutation warping achieves high efficiency with great flexibility on general machines. Special purpose machines cannot offer this flexibility in shading, combining, and filter choices. Changing viewpoints has been thought to be inefficient, and low quality filters in shearing methods have been used for efficiency or data duplication used for ray tracing, but our algorithm provides efficiency with a zoh, foh (trilinear filter), and generalizes to better filters. A three dimensional decomposition was introduced that generalizes the work in pure shears [8][11][9], and improves our direct resampling approach

[15]. Our implementation on the MasPar allows rendering with changing viewpoints of five frames/second and two frames/second for higher quality trilinear reconstruction on 128x128x128 byte volumes. This improves on previous results [2][3][9][10][12] because of the better filters used, and we discussed the filter differences.

Straight forward extensions to our techniques include coherency adaptations such as adaptive ray termination and adaptive quadrature. Perspective projection can be added as a follow on warp to the permutation warp for two passes versus four of [12]. The MasPar implementation can also be modified to up sample or down sample, least expensively as an up sample of the transformed image or down sample while warping. Our new decomposition will be useful for 6 pass pure shear multipass approaches versus eight passes [9].

Acknowledgments

We thank Professors David Salesin, Tony DeRose, and Steve Tanimoto for their help and feedback. Thanks to Professor Steve Mann for answering all kinds of graphics questions. MasPar engineers Jon Becher and Rex Thanakij provided valuable machine support as did Helen Asher who coordinated the MasPar Challenge contest by which a 16k processor MasPar was made available. We would also like to thank several anonymous referees.

Bibliography

[1] Blank, Tom The MasPar MP-1 Architecture. In *Proceedings of Compcon 90* (Feb. 26-Mar. 2, 1990), 20-24.

[2] Cameron, G.G. and P.E. Undrill. Rendering Volumetric Medical Image Data on a SIMD Architecture Computer. In *Proceedings of the Third Eurographics Workshop on Rendering*, (Bristol England, May 1992).

[3] Drebin, Robert A., Loren Carpenter, and Pat Hanrahan. Volume Rendering. In *Computer Graphics* 22, 4 (Aug. 1988), 65-74.

[4] Kaufman, Arie, Editor. *Volume Visualization*. IEEE Computer Society Press, Washington, D.C. 1991.

[5] Kruskal, Clyde P., Larry Rudolph, and Marc Snir. The Power of Parallel Prefix. In *Proceedings IEEE International Parallel Processing Symposium* (1985). 180-185.

[6] Montani, C., R. Perego, and R. Scopigno. Parallel Volume Visualization on a Hypercube Architecture. In *Proceedings of 1992 Workshop on Volume Visualization* (Oct. 1992), 9-16.

[7] Nieh, Jason and Marc Levoy. Volume Rendering on Scalable Shared-Memory MIMD Architectures. In *Proceedings of 1992 Workshop on Volume Visualization* (Oct. 1992), 17-24.

[8] Paeth, Alan W. A Fast Algorithm For General Raster Rotation. In *Proceedings Graphics Interface* (May 1986), 77-81.

[9] Schroder, Peter and James B. Salem. Fast Rotation of Volume Data on Data Parallel Architectures. In *Proceedings IEEE Visualization '91*, (San Diego, CA Oct. 1991), 50-57.

[10] Schroder, Peter and Gordon Stoll. Data Parallel Volume Rendering as Line Drawing. In *Proceedings of 1992 Workshop on Volume Visualization*, (Oct. 1992), 25-32.

[11] Tanaka, A., M. Kaneyama, S. Kazama, and O. Watanabe. A Rotation Method For Raster Image Using Skew Transformation. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition* (June 1986), 272-277.

[12] Vezina, Guy, Peter A. Fletcher, and Philip K. Robertson. Volume Rendering on the MasPar MP-1. In *Proceedings of 1992 Workshop on Volume Visualization*, (Oct. 1992), 3-8.

[13] Westover, Lee. Footprint Evaluation for Volume Rendering. In *Computer Graphics* 24, 4 (Aug. 1990), 367-376.

[14] Wittenbrink, Craig M. and Arun K. Somani. Cache Tiling for High Performance Morphological Image Processing. In *CAMP 91, Computer Architecture For Machine Perception*, (Paris, France, Dec. 1991), 427-438.

[15] Wittenbrink, Craig M. and Arun K. Somani. 2D and 3D optimal parallel image warping. In *Seventh International Parallel Processing Symposium*, (Newport Beach, CA, April 1993), 331-337.

[16] Wittenbrink, Craig M. Designing Optimal Parallel Volume Rendering Algorithms. Ph.D. dissertation, University of Washington, 1993.

[17] Yoo, Terry S., Ulrich Neumann, Henry Fuchs, Stephen M. Pizer, Tim Cullip, John Rhoades, Ross Whitaker. Achieving Direct Volume Visualization with Interactive Semantic Region Selection. In *Proceedings IEEE Visualization '91*, (San Diego, CA, Oct. 1991), 58-65.

TABLE 1 16k Processor MP-1 128³ Rendering Times in Milliseconds, Rotation on Multiple Axes

Filter	Rotation Axes	0	20	40	60	80
FOH	x	434	482	496	505	533
	y	434	485	494	498	537
	z	435	499	508	512	543
	x, y, and z	434	502	508	521	607
ZOH	x	145	186	196	205	237
	y	145	187	192	199	237
	z	146	206	213	222	240
	x, y, and z	145	207	212	224	253

TABLE 2 16k Processor MP-1 Rendering Times in Milliseconds, for many volume sizes

Filter	vol size	Mean	Min	Max
Trilinear	32 ³	13.3	11.5	16.0
	64 ³	68.7	58.9	83.4
	128 ³	507	434	607
	256 ³	3998	3429	4771
ZOH	32 ³	7.68	6.47	9.16
	64 ³	30.4	22.3	39.8
	128 ³	208	145	287
	256 ³	1623	1118	2245

TABLE 3 16k Processor MP-1 Warping Only Times in Milliseconds

Filter	vol size	Mean	Min	Max
Trilinear	32 ³	12.2	10.4	14.9
	64 ³	66.7	57.0	81.4
	128 ³	5023	429	602
	256 ³	3977	3407	4750
ZOH	32 ³	6.59	5.38	8.08
	64 ³	28.4	20.4	37.9
	128 ³	203	140	282
	256 ³	1602	1097	2224