Accelerating Virtual Endoscopy

József Koloszár, Yi Jae-Young
Dept. of Control Engineering and Information Technology,
Budapest University of Technology and Economics
Pázmány Péter sétány 1/D
1117, Budapest, Hungary
kj212@hszk.bme.hu

ABSTRACT

Applying volumetric ray tracing in interactive visualization has always been limited by low rendering speeds. This paper presents methods for accelerating first-hit ray tracing based virtual endoscopy with negligible impact on image quality, which aim at improving empty-space traversal (tracing all rays to the colon wall and storing the results in a depth buffer) and shading (surface normal approximation at hit locations and simple Phong shading applied to obtain pixel color). The first method proposed accelerates empty-space traversal by exploiting inter-ray coherence based on the fact that the colon wall is a (C^2) continuous natural surface, which does not exhibit erratic behavior, such as sharp jumps, steps or edges. The second method presented improves shading performance by using conditional interpolation in pixel space. Results have been successfully implemented in the virtual colonoscopy application ColVis, maintained by the author.

Keywords

Volume Visualization, Volumetric Ray Tracing, Virtual Endoscopy, Virtual Colonoscopy.

1. INTRODUCTION

Ray tracing based volume visualization is a direct volume rendering approach to visualizing various datasets. Common applications include medical visualization problems, such as virtual endoscopy, or more specifically virtual colonoscopy [Vis96]. Virtual Colonoscopy aims to reconstruct internal views of the human colon from CT (Computer Tomography) scans, preferably at interactive speeds, thus providing a diagnostic examination of the inside of the colon. It is argued to be a valid cost-effective and more patient comfortable alternative to classic colonoscopy in screening for colorectal polyps and cancerous tumors. In order for virtual colonoscopy to become a widespread diagnostic method, it is important that it not be confined to the realm of high-end graphic workstations [Li99] [Wan00].

Attaining interactive rendering performance - ten or more frames per second - is no trivial task, and has been subject to extensive research. Most methods suggested in the past where indirect, surface rendering based techniques. Though providing superior image quality, better accuracy, and requiring very little or no preprocessing, direct volume rendering algorithms lack the extensive hardware acceleration support widely available for surface based rendering. When implementing ray tracing based applications on mainstream PC hardware, performance is still the biggest issue. As will be shown, however, recent advances in computer hardware have made direct volume rendering a viable alternative [Joc02].

ColVis was developed to be a simple, convenient visualization tool, originally intended for virtual colonoscopy. Because of its very generic and direct approach, however, it proved to be applicable to other endoscopy problems, such as bronchoscopy.

Section 2 gives a brief overview of the ray tracing problem specific to virtual endoscopy, and the concept of separating empty-space traversal from shading is introduced. Sections 3 and 4 review previous techniques, and present the new algorithms for empty-space traversal and shading respectively. The numeric results from benchmarking are summarized in Section 5. Results are very promising, and indicate that virtual endoscopy using direct volume rendering feasible on mainstream PC hardware.

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

2. RAY TRACING IN VIRTUAL ENDOSCOPY

When dealing with virtual endoscopy, the rendering problem can be defined very clearly. Further discussion is limited to very basic virtual colonoscopy, and the following assumptions can be made without penalty [Joc01]:

- The camera is always inside the colon.
- The colon wall is defined by a threshold value, and all voxels inside the colon have a lower value.
- The inside of the colon is considered non-interfering empty-space (air).

The third assumption is justified in theory, as the colon is cleansed and inflated prior to scanning. In practice, the cleansing is not perfect. However, differentiating the colon wall from occasional fecal residue (fecal tagging) lies outside the scope of this paper.

What the above assumptions mean in terms of rendering is that rays cast from the camera position traverse empty-space without any calculations, hit the colon wall, where a surface normal is estimated, which is used to obtain the shade, or color of the pixel corresponding to the ray. Note that the above is a simple case of direct volume rendering, where the opacity curve is defined as a simple step function at the threshold specified. Regarding visual quality, the requirement is that features greater than 5mm should not be lost, as polyps of this size should be recognizable.

Two stages of rendering are distinguished, and discussed separately. In the first stage, empty-space was traversal, all rays are traced to the colon wall and the results are stored in a depth buffer. In the second stage, shading, normal vectors are approximated at hit locations and simple Phong shading is applied to obtain pixel color. The lighting model consisted of a single camera-mounted light source (headlight model).

3. EMPTY-SPACE TRAVERSAL

A number of methods have already been suggested to accelerate empty space traversal in volume rendering. The most notable ones are potential-field based, or exploit frame-to-frame coherence. In the potential field method [Wan99], the distance to the colon wall is calculated and stored for every voxel inside the colon. This value is a safe increment to the ray in every iteration of the trace. One of the drawbacks to this method is that the setup is sensitive to the threshold value defining the colon wall, and the recalculation of the potential field buffer cannot be performed at interactive speed.

Another example of buffer based methods is the surface-volume rendering hybrid [You97]. In this case the "buffer" is, in fact, the extracted colon surface data. Surface rendering is used to obtain the z-buffer only, and shading is done using methods of volume rendering for superior image quality. As surface based depth rendering has hardware acceleration support, this is probably the fastest way to traverse empty space. However, pre-calculations are even more expensive and lengthy than setting up a potential field, memory issues are also a concern as are other non-trivialities of surface rendering [Wan99].

Exploiting frame-to-frame coherence [Vil99] assumes little camera displacements and rotation between frames, but information derived and used in the acceleration process is less precise [Joc01]. The following technique can be used instead of or to complement either of the above methods.

Proposed Algorithm

With no acceleration all rays are cast from the camera position, and are incremented by unit length every iteration of the trace (voxel precision). The maximum value of the eight voxels cornering the sampling point is checked against the threshold value. If the result indicates a possible hit, a refined calculation is performed to scan for the hit location every 0.2 units along the ray (sub-voxel precision) and checking against the value obtained by tri-linear interpolation from the eight cornering voxels. To improve collision detection performance, the maximum value of the cornering voxels is pre-calculated for every cell (unit volume cornered by eight neighboring voxels), reducing the number of comparisons from eight to one per sampling. The calculation is performed only once when the volume is first loaded, and is application independent. The buffer is of the same size as the original volume.

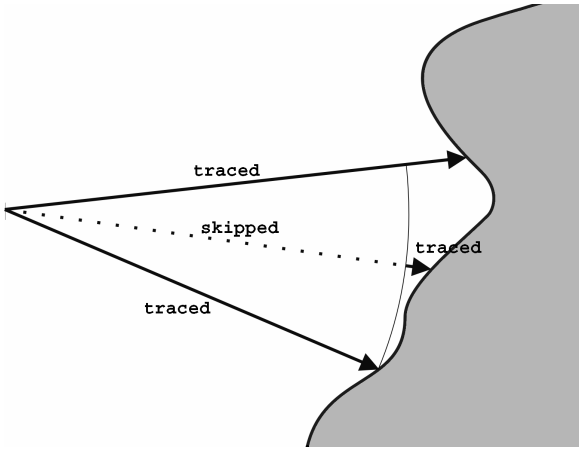

Figure 1: Exploiting inter-ray coherence in empty-space traversal.

Acceleration is based on the fact that the colon wall is a (C^2) continuous natural surface, which does not exhibit erratic behavior, such as sharp jumps, steps or edges. Thus if two close rays have already been traced to the colon wall, a ray between them would probably not hit the surface closer to the camera than the shorter one of the two. The exception would be hitting the "peak" of a concave curvature. This case could be detected either by evaluating the normals at the two hit locations already obtained, or by checking the result of the trace for the occurrence of an unusual jump in z-space (depth image). However, reducing full traces to as low as only 64x64 rays results in only negligible loss of image quality in the focus of attention (features closer than 100 units).

This method provides a dramatic increase in empty space traversal performance. Reducing full traces to one fourth results in about 60% increase in performance.

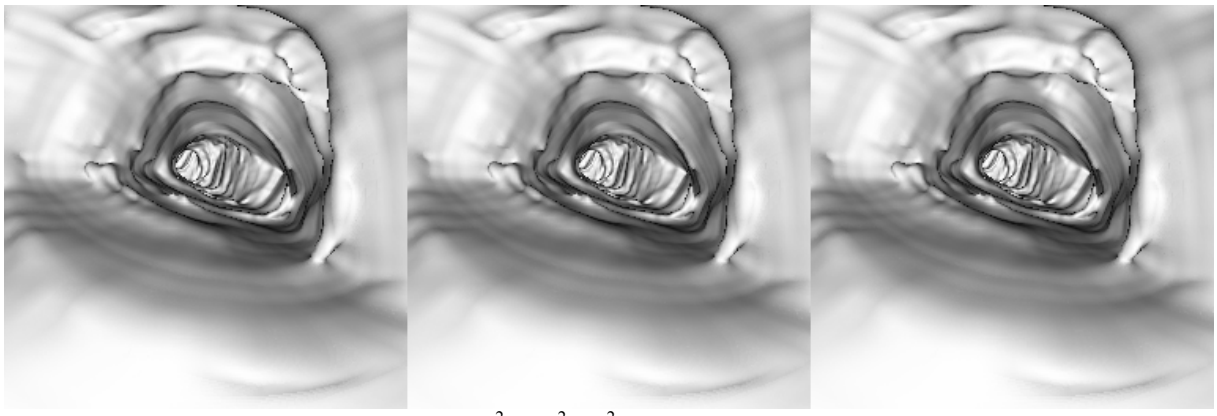

Figure 2: The same scene traced with 256^2, 128^2, 64^2 (from left to right respectively) full traces traversing empty-space.

4. SHADING

The real issue in determining pixel color is approximating the surface normal at the hit location, which is passed to the shader. Volume data in case of virtual colonoscopy is obtained by the sampling of a continuous object, the human body. Through discretization the information, from which exact normals could be derived, is lost, and approximating techniques must be used. Discrete normal estimation methods usually fall into either of two fundamentally different categories: image-space and object-space methods [Neu00].

Image space methods base their approximations on the 2D projected image. A well-known image-space method, depth gradient shading calculates normals from gradients in the z-buffer. Approaches like this, however, fail to deliver expected image quality in most virtual endoscopy applications without some form of tuning to reduce artifacts.

Object-space methods, on the other hand, work with the 3D neighborhood of the point being sampled. Methods in this category are fairly diverse and are based on concepts that range from traditional derivative filters to associating voxels with convex geometric primitives or local iso-surface extraction. Mathematically accurate approximating methods tend to involve solving systems of equations or finding the roots of higher degree polynomials. Most of these methods are either time consuming or do not deliver the expected image quality. Therefore, the implementation of a relatively new method based on 4D linear regression [Neu00] was chosen to estimate the gradient (which can be normalized and used as the surface normal). The concept being that a linear function

$$f(x,y,z) = Ax+By+Cz+D$$

is used to estimate density in the close proximity of a voxel where *x, y* and *z* are distances from the voxel along the corresponding axis. Minimizing the error-squared based on values of the 26 closest neighboring voxels *A, B, C* and *D* can be calculated very efficiently. The *(A,B,C)* vector is a good estimate of the gradient and *D* can be used as the filtered density value for the pixel.

The algorithm performs a linear regression based approximation of the gradient vectors at the eight voxels cornering the hit location. These gradients are in turn subjected to tri-linear interpolation to estimate the gradient at the sampling point, which is normalized to yield the surface normal approximation used for shading the hit-location. This method was found to deliver great image-quality with sufficient precision.

Proposed Algorithm

In [Kol02] calculations for minimizing the error-squared were still deemed costly, and a hybrid rendering engine was proposed to further boost performance. The hybrid used a z-space based gradient estimation method, which performed much faster than the algorithm detailed above. In the implementation only regions close to the camera were rendered with the linear regression based algorithm, while the rest was shaded with the z-buffer based method, resulting in 30% increase in performance at the price of a noticeable hit to image quality.

Recent optimizations by restructuring code have drastically accelerated the linear regression calculations. As a result the two shading methods used in the hybrid now perform at comparable levels, making the concept of the hybrid obsolete for the time being.

To accelerate the algorithm conditional interpolation in pixel-space is proposed. Note that with the lighting model specified pixel color is in direct relation with the angle between the surface normal and light direction, in our case with the direction of the camera. Assuming that two close pixels have already been rendered, a pixel between them must not necessarily be shaded from scratch. Two conditions are checked to determine whether a simple interpolation between the two pixels already rendered is enough to estimate the value for the new pixel:

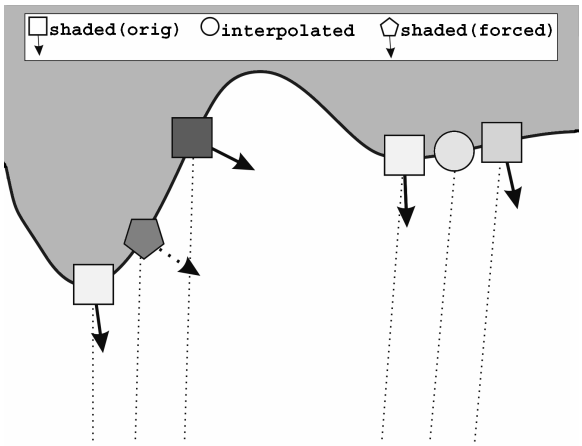

Figure 3: Conditional interpolation in shading.

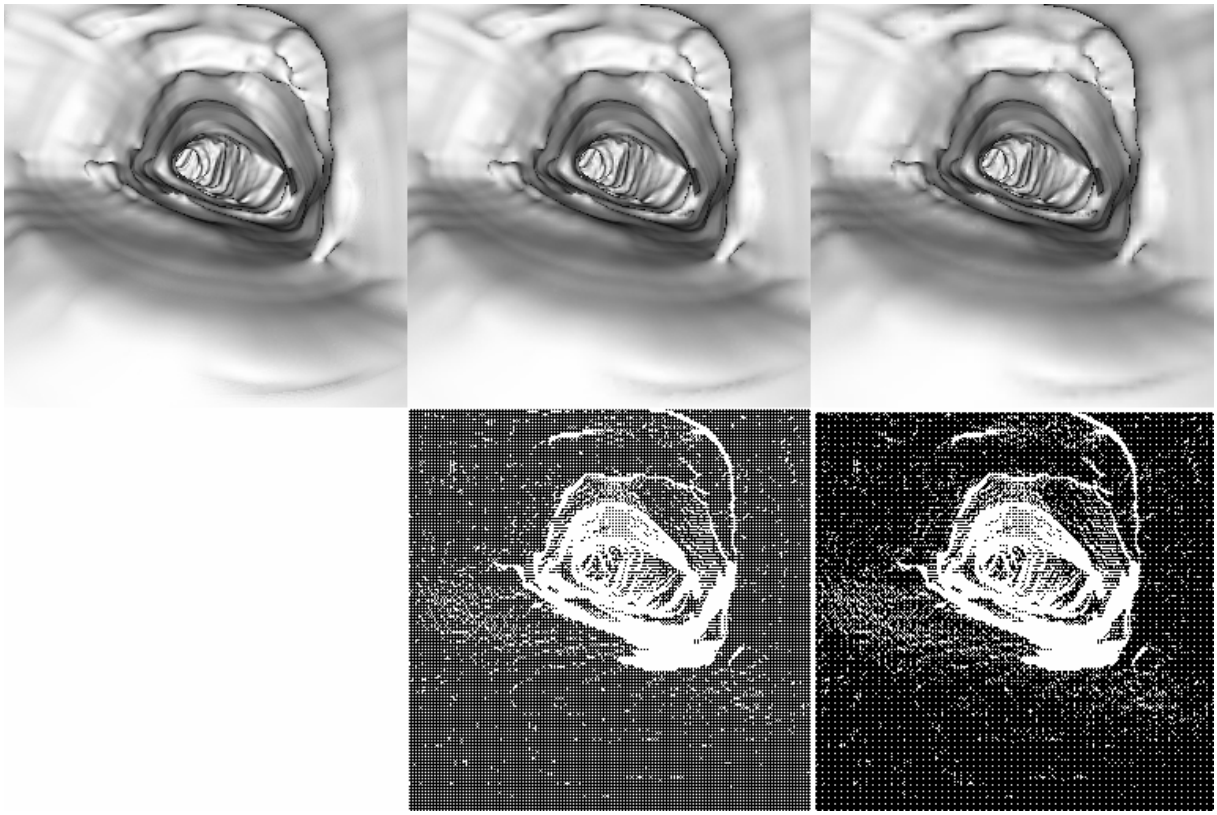

Figure 4:

Top: The same scene shaded from 256^2, 128^2, 64^2 (from left to right respectively) pixels shaded, with the rest interpolated conditionally using.

Bottom: Bitmaps indicating shaded (white) and interpolated (black) pixels for the corresponding images above. The ratio of pixels interpolated is 0%, 62% and 74% (from left to right respectively).

The first is an absolute condition. Interpolation is safe if both already rendered pixels are "bright enough", meaning that the surface region being rendered is close to "facing" the camera.

The second, relative condition checks the difference in depth between the two already rendered hit location. If they differ too much, interpolation is deemed unsafe. If either check fails, the pixel in question is shaded from scratch.

The quality-performance compromise of the acceleration can be tuned by adjusting the parameters defining the conditions. We have found that rendering every fourth pixel from scratch and conditionally interpolating the rest results in around 65% increase in performance in the average look-forward view.

5. RESULTS

All results presented in this section were obtained from implementing on the following system:

Platform: 1.6GHz PC (x86) w/ 768MB RAM running MS Windows XP Pro.
Implementation: C++ with DirectDraw compiled using Visual C++. No low-level, hardware-specific code optimizations have been implemented, all code except drawing (DirectDraw) and window handling (Win32SDK) in ANSI C++.
Dataset: 512x512x192 array of voxels stored at 16bit precision. Benchmarking was performed at 256x256 and 128x128 rendering resolutions.
The Tables below show results for empty space traversal and shading of images. The parameters for conditional interpolating in shading were specified as follows: 96/256 for "brightness" in the absolute, and 1.0 in the "depth delta" relative checks.

In each test the camera was positioned in the same (typical) location and rotated a complete 360-degree circle, while tilting up and down in a wave-like manner to obtain 180 frames. Timing was implemented in the code on a per-frame basis, the results accumulated and averaged.

Traced Initially	256x256	128x128	64x64
Time[ms]	336	144	103

Table 1: Average empty-space traversal times for 256x256 image.

Shaded Initially	256x256	128x128	64x64
Time[ms]	177	53	25

Table 2: Average shading times for 256x256 image.

Traced Initially	128x128	64x64	32x32
Time[ms]	84	34	29

Table 3: Average empty-space traversal times for 128x128 image.

Shaded Initially	128x128	64x64	32x32
Time[ms]	45	14	8

Table 4: Average shading times for 128x128 image.

Sh\Tr	256x256	128x128	64x64
256x256	1,949	3,115	3,56
128x128	3,115	5,116	6,442
64x64	3,583	6,507	7,95

Table 5: Overall average performance for 256x256 rendering with varying levels of acceleration in shading (columns) and empty-space traversal (rows) in frames per second (fps).

Sh\Tr	128x128	64x64	32x32
128x128	7,798	12,252	13,575
64x64	10,16	19,386	23,997
32x32	10,66	21,763	27,236

Table 6: Overall average performance for 128x128 rendering with varying levels of acceleration in shading (columns) and empty-space traversal (rows) in frames per second (fps).

The tri-linear interpolation based collision detection used to find the exact hit location at sub-voxel precision poses the limit to accelerating empty-space traversal, as associated calculations must not only be performed for "full traces" (rays initiated from the camera position), but every ray in the image. Comparing the results from tracing all rays from the camera in the low resolution image (128x128) with those obtained while rendering at higher resolution (256x256) with just every fourth ray traced from the camera, one can conclude that around $1/4^{th}$ of the time tracing each ray is spent in the sub-voxel precision stage. Volumes are generally sampled with voxels not representing unit cubes, but square based slabs instead. As resampling the volume would raise memory issues in most cases (boosting volume data size by a factor of up to 4-8, depending on the original CT scanning parameters), the engine presented resolves the issue by performing additional on-the-fly interpolation volume data is referenced. Future optimizations will target the resulting performance hit as well as the relative high cost of the sub-voxel sampling stage.

Calculations involved in the surface normal estimation method by applying tri-linear interpolation to the gradients estimated using linear regression at the eight voxels cornering the hit location have undergone some manual optimization. In the original code calculations were performed by the formula as presented in [Neu00]. By unrolling nested loops, rearranging and eliminating instructions overall performance was effectively doubled. In case of the 256x256 image, with 64x64 rays cast and 64x64 pixels shaded initially, the average frame rate increased from 4.09 to 7.95 fps.

It was also found that implementing some low-level optimizations, such as hand-coded vector arithmetic using *3Dnow!* did not drastically improve overall performance over the binary generated by the compiler. Though it has not been the focus of this study, the author suggests that this mainly due to the true low-level bottleneck being memory access, or rather the lack of cache hits, as data was stored in unstructured linear multidimensional arrays.

6. CONCLUSIONS

It was found that conditional interpolation based techniques applied in object-based ray tracers can also be used to efficiently accelerate volumetric ray tracing. Results suggest that computing power of mainstream processors today can cope with raw empty-space traversal and high-quality shading at interactive, but not yet at true real-time (>20) frame rates.

Full traces can be reduced to and beyond 64x64 rays, without loss in visual quality or detail on significant features. Depth buffers can be set up in under 30ms for low resolution (128x128), and around 110ms for higher resolution (256x256) images. Though still the bottleneck in the rendering engine presented, empty-space traversal will cease to be a significant issue in virtual endoscopy in the very near future.

Despite the impressive capabilities of available C++ compilers (Microsoft Visual C++, GCC, Intel C++), it was found that they still lag behind in some crucial aspects of optimizing volume rendering related code. Though an extensive evaluation of compilers was not the main focus of this research, a number of compilers and compiler options have been experimented with. The dramatic performance gain resulting from high-level manual optimization, however, suggests that compilers lack in efficiency when it comes to unrolling and optimizing (multi-dimensional) nested loops with otherwise simple bodies, or simplifying and/or eliminating floating-point arithmetic expressions.

With over 10 frames per second possible, rendering images of acceptable resolutions providing enough detail on significant features, it is concluded that first-hit volumetric ray tracing based virtual endoscopy, virtual colonoscopy in particular is technically feasible and application-ready on mainstream PC hardware, even without extensive low-level optimizations. Further performance increases are expected once volume data storage is optimized to maximize cache hits, and SIMD instructions are manually implemented.

7. ACKNOWLEDGMENT

This project is supported in part by “Ötletlabor” project, by the Scientific Research Fund (OTKA ref. No: T029135) and by the Slovene-Hungarian Action Fund.

8. REFERENCES

[Chi98] R. Chiou, A. Kaufman, Z. Liang, L. Hong, M. Achniotou: Interactive Path Planning for Virtual Endoscopy, Conf Record IEEE NSS-MIC, Nov. 1998.

[Chi99] R. Chiou, A. Kaufman, Z. Liang, L. Hong, and M. Achniotou: Interactive Fly-Path Planning Using Potential Fields and Cell Decomposition for Virtual Endoscopy. IEEE Trans Nuclear Sciences, vol. 46, no. 4, Aug 1999, pp. 1045-1049

[Csé98] Csébfalvi, B. & Szirmay-Kalos, L.: Interactive Volume Rotation. Machine Graphics and Vision, Vol. 7, 1998. p 793–806.

[Hon97] L. Hong, S. Muraki, A. Kaufman, D. Bartz, T. He: Virtual Voyage: Interactive Navigation in the Human Colon, Proc. ACM SIGGRAPH '97, Aug. 1997, pp. 27-34

[Kol02] Koloszar, J. & Jocha, D.: Accelerating Volumetric Ray Tracing in Virtual Endoscopy. First Hungarian National Conference on Computer Graphics, 2002.

[Joc01] Jocha D. & Koloszár J.: Interactive Virtual Colonoscopy. CESCG 2001, Bumerice, SK.

[Joc02] Jocha D. & Koloszár, J.: Virtual Colonoscopy using Direct Volume Rendering of Surfaces from Volumetric Data. KEPAF 2002, Hungary, 2002.

[Li99] Li W. Li, M. Wan, B. Chen, and A. Kaufman (1999): Virtual Colonoscopy Powered by VolumePro. pp., 1999.

[Neu00] L. Neumann, B. Csébfalvi, A. König, E. Gröller: Gradient Estimation in Volume Data using 4D Linear Regression. EUROGRAPHICS 2000, Eurographics Association, 2000.

[Szi99] Szirmay-Kalos, L.: Számítógépes Grafika. Computerbooks, 1999.

[Vil99] A. Villanova I Bartroli, K. Buhler: On Coherence in Rendering. Frorshungsseminar Visualisierung WS98/99, 1999.

[Vis96] A. Viswambharan, M. Wax, L. Hong, A. Kaufman, Z. Liang, T. Botton: Virtual Colonoscopy: Three-dimensional Reconstruction of the Mucosal Surface of the Colon. Conf of Radiological Society of North America (RSNA), Dec. 1996, pp. 565 (Scientific Merit Award)

[Wan99] M. Wan, Q. Tang, A. Kaufman, and Z. Liang: Volume Rendering Based Interactive Navigation within the Human Colon. IEEE Visualization ’99 conference, San Francisco, CA, Oct, 1999, pp. 397-400

[Wan00] M. Wan, W. Li, K. Kreeger, I. Bitter, A. Kaufman, Z. Liang, D. Chen, and M. Wax: 3D Virtual Colonoscopy with Real-time Volume Rendering. SPIE Medical Imaging 2000, 2000.

[You97] S. You, L. Hong, M. Wan, K. Junyaprasert, A. Kaufman, S. Muraki, Y. Zhou, M.Wax, and Z. Liang: Interactive Volume Rendering for Virtual Colonscopy. IEEE Visualization ’97 Conf Proc, ACM/SIGGRAPH Press, Oct. 1997, pp. 433-436