Appeared: IEEE Visualization '98, Oct 18-23, 1998.

Efficient Warping for Architectural Walkthroughs Using Layered Depth Images

Voicu Popescu, Anselmo Lastra, Daniel Aliaga, Manuel de Oliveira Neto

University of North Carolina at Chapel Hill

Abstract

This paper presents efficient image-based rendering techniques used in the context of an architectural walkthrough system. Portals (doors and windows) are rendered by warping layered depth images (LDIs). In a preprocessing phase, for every portal, a number of pre-rendered images are combined into a LDI. The resulting LDI stores, exactly once, all surfaces visible in at least one of the images used in the construction, so most of the exposure errors are efficiently eliminated. The LDI can be warped in the McMillan occlusion compatible ordering. A substantial increase in performance is obtained by warping in parallel. Our parallelization scheme achieves good load balancing, scales with the number of processors, and preserves the occlusion compatible ordering. A fast, conservative reference-image-space clipping algorithm also reduces the warping effort.

CR Categories and Subject Descriptors: I.3.1 [Computer Graphics]: Hardware Architecture - Parallel processing; I.3.3 [Computer Graphics]: Picture/image Generation - Viewing Algorithms; I.3.8 [Computer Graphics]: Applications.

Additional Keywords: image-based rendering, parallel warping, occlusion compatible ordering for discrete images, portal, cell, exposure error, layered depth image, clipping, architectural walkthrough.

1 INTRODUCTION

In an architectural walkthrough, the scene is naturally divided into cells (rooms) linked by portals (doors, windows, etc.). A number of methods have been developed to compute which cells and which portals are visible in a certain view [Airey90]. However, in the case of complex models with a large number of geometric primitives, conventionally rendering all the visible cells is too slow to maintain interactive rates. The performance is considerably improved if the only primitives rendered are the ones in the current cell and each visible portal is rendered by warping a pre-rendered image [Aliaga97]. Important artifacts, called exposure errors, occur when a desired view exposes parts of the scene that are not represented in the image to be warped. To address this problem, *layered depth images* (LDIs) are used.

{popescu, lastra, aliaga, oliveira}@cs.unc.edu
Computer Science Department, Sitterson Hall, CB#3175
Chapel Hill, NC 27599-3175

LDIs [Gortler97] have, like ordinary images, a set of parameters that define the reference view. Unlike ordinary images, they store information about surfaces that are not visible in the reference view but might become exposed. This drastically reduces the occurrence of exposure errors. Since the portal that is rendered with the LDI may be viewed from many directions, the LDI is in general wider than it needs to be for any single viewpoint. A recursive clipping algorithm is used to reduce the size of the LDI that has to be warped.

While the conventional graphics hardware is rendering the geometric model of the current cell, the general-purpose processors of the system are available for warping. Any good parallelization scheme has to take into account that pixels are frequently warped into the same location of the desired image. Epipolar geometry in the context of 3-D warping [McMillan95A], [McMillan97] shows how points move in the image when they are warped. In order to understand which pixels may occlude each other, this theory has to be carefully adapted to the warping of discrete images where samples have an associated non-zero area. If different processors warp such pixels without due care, visibility errors will occur. We overcome these problems and our parallelization scheme achieves good load balancing and scales with the number of processors.

2 PREVIOUS WORK

An early technique used to speed up architectural walkthroughs was to compute the visible cells for every desired view and render only those cells [Airey90]. For models with a great deal of geometry, this technique alone is not sufficient.

To reduce the amount of rendered geometry, one can use textures at the portals [Aliaga97]. In order to get approximately correct images, a large number of textures have to be used. With few textures, this technique exhibits a "popping" effect, and a lack of motion parallax.

More recently, image warping was used to solve these two problems [Rafferty98]. A number of images with depth were pre-rendered from a certain number of viewpoints uniformly distributed on a semicircle in front of the portal. At run time, one or two such images are warped for every portal. Each image is warped in *occlusion compatible order* [McMillan95B], a way of ensuring correct visibility without depth comparison. When a single image is warped, exposure errors are common. Unfortunately, McMillan's algorithm does not guarantee correct visibility when multiple images are warped to the desired view. Furthermore, warping two images incurs redundant work since many of the samples are identical.

The use of layered depth images [Gortler97, Max95] is an elegant solution to many of the problems of the previous methods. Exposure errors are drastically reduced, and the more images that are used to build the LDI, the less likely it is that such errors occur. Also, the amount of work done when warping a LDI is

comparable to the amount of work required to warp a regular image extended with depth; LDIs can be warped in occlusion compatible order, therefore requiring no depth comparison at run time.

3 ARCHITECTURAL WALKTHROUGH SYSTEM

Our architectural walkthrough system is similar to that described by [Rafferty98]. The current cell is rendered using geometry, but distant portals are rendered using images, in our case LDIs. When the viewpoint approaches a portal, the next cell is rendered using geometry. Thus, geometry is used for nearby objects, while more distant ones are rendered from images.

3.1. Constructing the LDIs

A LDI is constructed as a preprocessing step. Consider a portal, and a semicircle in front of it (Figure 1). The first step is to render 2n+1 images with depth, with their centers of projection (COP) equally spaced along the semicircle and with the view oriented toward the center of the portal. Figure 1 depicts a situation for n = 6.

First, the central image, indexed 0, is stored in the LDI. Then, the remaining twelve images, in order from 1 to 12, are warped to the plane of the central image. If a sample lands at an empty location, it is stored, and its generalized disparity (equivalent to depth) [McMillan97] with respect to the LDI is computed and stored; if, however, the sample lands at an occupied location, it is stored only if it represents a different surface. In order to decide whether the surfaces are distinct, the range values of the two samples are compared. If the difference in range is greater than some threshold, the sample is stored in a new layer. If the range difference is not large enough, but the colors are different, the new sample is also stored in a new layer. This ensures that enough samples are stored in the LDI for the surfaces that are not well sampled by the central view (the LDI view). The cost is the poor filtering of these surfaces when seen from views close to the central view.

The construction of LDIs is prone to all errors inherent in the warping operation: the reconstruction is not perfect, and the visible samples are forced to land at integer coordinates in the desired image. These errors are amplified by the warping of the LDI. Under the assumption that the portal's desired view parameters are usually close to the reference view parameters, such errors are minimized when the images close to the construction view are used first in the construction of the LDI.

When the portal is frequently viewed at acute angles, it would be worthwhile to construct LDIs viewing the portal from different angles. The trade off for the gain in quality is additional storage.

3.2. Warping LDIs

Although a LDI stores information about hidden surfaces, it behaves like a single image when warped, since there is only one set of view parameters. This section shows how the occlusion compatible ordering algorithm can be adapted to LDIs, and describes a parallel implementation for it.

3.2.1. Occlusion Compatible Traversal

As [Gortler97] observes, occlusion compatible traversal works correctly for LDIs if the locations of the LDI are visited in the order described by [McMillan95B] and the layers are warped in back to front order.

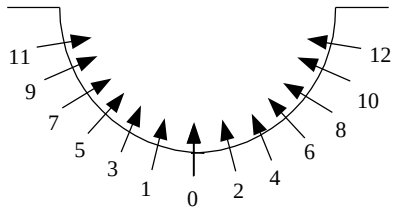


Figure 1: *View vectors of images used to construct a LDI.*

To see that this is true, it is necessary to examine two cases. The first case occurs when samples from two LDI locations (at any layers) warp to the same location in the final image. This is equivalent to warping a single-layer image with samples at the same depth as those in the LDI. [McMillan96] proves this correct. The second case, samples warped from different layers at the same location of the LDI onto the same pixel in the final image, clearly preserves visibility if traversed in back-to-front order. Notice that for a walkthrough application, LDIs are never seen from the back.

3.2.2. Problems for Parallelization

Pixels in discrete images have a *non-negligible area.* This, and the fact that the desired image is also discrete, presents practical difficulties that are not apparent in the continuous domain of epipolar lines and point samples. Let us define the epipolar extent of a pixel as the set of epipolar lines that intersect the pixel (Figure 2). The projections of two pixels will not intersect in the desired image if the epipolar extents of the pixels are disjoint. Otherwise, the warped pixels may occlude one another and must be rendered in visibility preserving order. Figure 3 shows the visibility relationships (ordering of pairs of pixels) for a positive epipole.

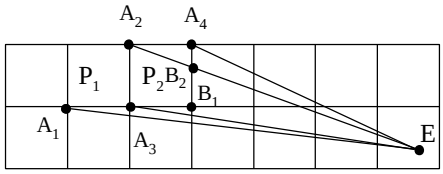


Figure 2: *The angle A_1EA_2 is the epipolar extent of pixel P_1. The epipolar extents of P_1 and P_2 are not disjoint.*

McMillan's paper proposed a way of splitting the reference image into 1, 2 or 4 sheets that are traversed incrementally in either row- or column-major order. This method must be modified to take into account the fact that the epipole will often be located within the area of a pixel. Problems occur with the pixels on the row and column of the epipole (for example, P_NE in figure 3). They have to be warped either first or last, depending on the sign of the epipole. This can be done by making sure that the sheets are warped in the right order (Figure 4). If we warp the sheets independently (for example, simultaneously processing them on different processors), then the epipole's row and column must be warped separately before or after the sheets, again according to the epipole's sign.

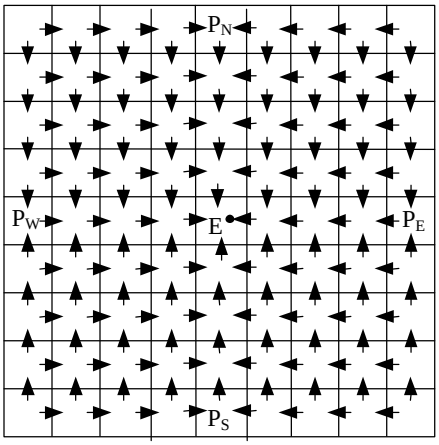


***Figure 3**: The epipole falls inside a pixel. The arrows show the visibility relationship between pixels.*

How big is the potential error when the rendering order of two pixels is wrong? It is the area of intersection of the two projections. The potential error for the example in figure 2 is proportional to the area of quadrilateral $A_2A_3B_1B_2$. Therefore, in figure 3, the smallest potential errors will occur when the warping order is wrong in rows and columns, such as those containing E and P_E, P_S, P_W, P_N. The largest potential errors will occur when using the wrong order near diagonals in the reference image. This fact makes correct and load-balanced parallelization difficult.

3.2.3. Parallel Warping in Occlusion Compatible Order

When warping in parallel, we would like to achieve the following four goals: use visibility-preserving order, balance the workload, preserve the advantages of locality and incremental computation exhibited by serial warping, and use all available processors. In [Rafferty98], each of the four sheets was assigned to a different processor. This approach produces good load balancing only when the sheets are of comparable size, that is, when the epipole falls in the center of the reference image. However, this happens only rarely. In fact, we often see only two sheets. Another limitation is that no more than 4 processors can ever be used.

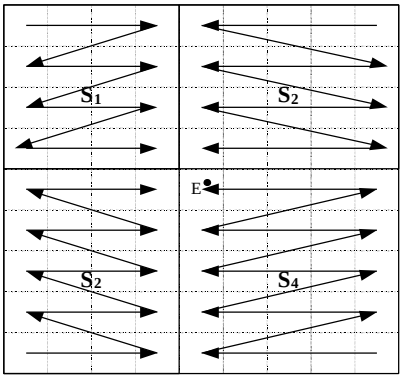


***Figure 4**: In this case S_1 has to be warped first then any of the two S_2 and then S_4. S_1 has no pixel from the row or column of the epipole, they belong to the S_2's; similar for S_2 and S_4.*

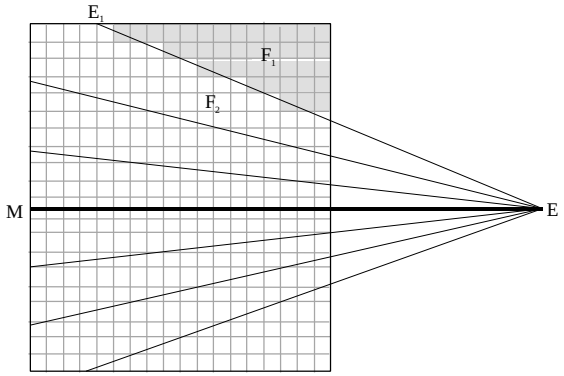


***Figure 5:** In this case the epipole E falls outside the reference image (or the LDI). The horizontal epipolar line EM divides the image into two sheets. Using 3 epipolar lines, each of the sheets is divided into 4 fragments (one for each available processor) of equal area. The lightly shaded pixels form fragment F_1. The epipolar line EE_1 belongs alternatively to the fragments F_1 and F_2 so the pixels that it traverses have to be warped in visibility preserving order. This cannot be enforced when the two fragments are warped in parallel and visibility artifacts will occur.*

This subsection describes a method that achieves much better load balancing and scales well. Each of the sheets is split into *p* fragments of equal area, where *p* is the number of available processors (Figure 5). The sheets are split along epipolar lines in order to avoid as many concurrent writes as possible. The resulting fragments are triangles or convex quadrilaterals that are stored as a collection of scan lines that are then efficiently warped. Depending on the number of sheets, there are *p, 2p*, or *4p* fragments to be warped. Splitting the entire image into *p* fragments is a less appealing solution because fragments that belong to two sheets have to be traversed in two different traversal orders.

Unfortunately, visible artifacts occur along the borders of the fragments (see color plate 4). Pixels that share an epipolar line should be warped in a well defined order (according to the sign of the epipole). This cannot be enforced for pixels that belong to different fragments since they are warped on different processors. Here is the solution we propose (see figure 6 and color plate 4):

1. Shrink the fragments slightly, so that no epipolar line crosses more than one fragment. This means that no pixel from one fragment can warp into another fragment. The pixels between the fragments form a *buffer zone.* Warp the shrunken fragments in parallel.

2. By now a substantial part of the desired image is ready. Compute the pixels of the desired image that might not be correct. They are the ones onto which the buffer zone pixels of the reference image may warp.

3. For each buffer zone, compute a corresponding *extended buffer zone* (Figure 6). It contains all of the pixels that might potentially warp to the same locations as pixels from the buffer zone. Warp the extended buffer zones, in parallel, in occlusion compatible ordering but *allow writing only to the potentially wrong pixels* (computed at step 2).

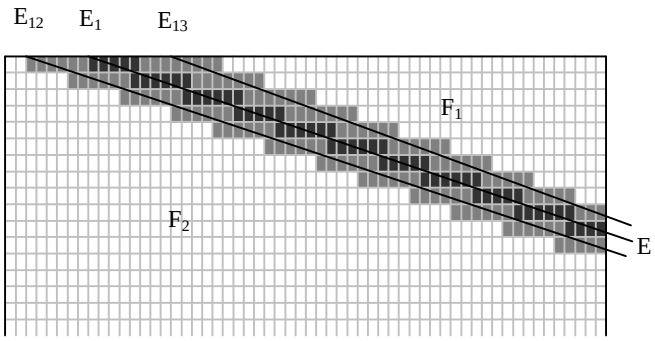


***Figure 6**: The pixels at the border between the fragments are warped in arbitrary order, which produces visibility artifacts. To avoid this problem, the fragments are shrunk by creating a buffer zone (dark shade in the figure). The pixels that might interfere when warped with the pixels in the buffer zone form the extended buffer zone (light shade in the figure). The extended buffer zones are computed using the epipolar lines EE_{12} and EE_{13} that enclose the buffer zone. The extended buffer zones are warped as described in the paper to complete the new image.*

This two-pass approach warps some of the pixels twice, but they form only a very small part of the reference image.

3.2.4. Conservative Reference Image Space Clipping of LDIs

In the walkthrough application, we construct LDIs by combining 2*n+1 images from different viewpoints. The resulting horizontal field of view is, in general, quite large. Usually, for a desired view, only part of the horizontal field of view is needed. Eliminating, before warping, the columns of the LDI that cannot be seen in the desired view of the portal brings a substantial increase in performance at a very low cost.

In order to do this, a binary tree is built recursively. The root stores the maximum and minimum disparities of the entire LDI. Then the LDI is divided into two equal parts, with a vertical line. The child at each branch of the tree always stores the extreme disparities of the corresponding half of the LDI stored at its parent. The leaves of the tree correspond to each column of the LDI, while every node corresponds to a group of contiguous columns.

At run time, the columns of the LDI that cannot possibly be visible are eliminated using a recursive clipping algorithm that uses the minimum and maximum disparities stored in the binary tree. In order to decide if a rectangular region from the reference image is visible, its four corners are warped with the minimum and maximum disparities of the region. A rectangular bounding box is computed for the eight resulting points. If it does not intersect the desired projection of the portal, it is safe to not warp the rectangular region.

4 IMPLEMENTATION AND RESULTS

The system was implemented on a Silicon Graphics Onyx^2 (four R10000 processors) with Infinite Reality graphics. The system is coded in C++ and uses the OpenGL graphics library. The architectural model used is that of large one-story house modeled using 528,000 polygons.

The LDIs were constructed in a preprocessing phase. Thirteen textures with depth were rendered for every LDI. They are 256 by 256 pixels in size with 24 bits for color and a floating-point disparity value. The thirteen textures were warped to the construction view of the LDI that, in our experiment, was perpendicular to the center of the portal. The seventeen LDIs created for the model had maximum depth between 4 and 16. One LDI is generated in less than 30 seconds and the process is fully automatic. Table 1 shows that in a typical LDI most of the samples are at level 1 and fewer than 1% are at depth 4 and above. The pixels that have samples in a large number of layers are due to scene surfaces that are aligned with a ray from the reference view of the LDI. Warping such pixels is not less efficient. On the contrary: only a small part of the warping equation needs to be evaluated for the various samples. The average size of the files that store the LDIs is about 2.5 MB. In addition to the color and depth samples, the files store the number of layers at each location.

	Abs	%
Total Number of samples	189,688	100
Samples in layer 1	150,080	79.11
Samples in layers 2-3	38,049	20.05
Samples in layers 4+	1,559	0.84
Size of LDI (pixels layer 1)	536x280	

***Table 1:** Distribution of samples in the layers of a typical LDI.*

The preloaded LDIs are warped whenever their corresponding portal is visible. First the recursive-clipping algorithm is run using the binary-disparities tree computed at load-time. On average the LDIs are clipped to 50% of their initial size when the entire portal is visible. The worst case for the clipping algorithm is when the portal is viewed from a close distance and at an oblique angle. Then the clipped LDI represents 75% of its initial size. The size of the clipped LDI decreases to 0 as the portal disappears from sight. On our system the recursive clipping takes less than 2 ms, insignificant when compared to the benefits it brings.

At run time parts of the LDI are traversed in row-major order, either left to right or right to left, the multiple locations always visited in back to front order. In order to improve the incrementality and locality of the warping, the samples of the LDI are saved in two arrays: the locations of the LDI are traversed in row-major order, but the layers of each location are saved once in back-to-front and once in front-to-back order.

The combination of the techniques presented was tested on a typical architectural walkthrough path. The *warping times* were reduced on average by a factor of 3.45 versus warping the LDIs serially. The speedup obtained over serially warping *a single regular image* was 1.92. The LDI, of course, drastically reduces the annoying exposure errors exhibited by single images. Over the sheet-based parallel warping of LDIs the speedup was 1.67. The average frame rate was 19 fps with a minimum of 3.5 fps (when more than 6 LDIs were present in the view frustum).

5 FUTURE WORK

The paths presented had the viewpoint moving on a horizontal plane. Future work will investigate the construction of the LDIs when the viewpoint is free to move anywhere in the model's volume. Determining how many LDIs are needed and where to

place them in order to guarantee a certain reconstruction quality is another open problem that deserves future efforts.

Another interesting question that remains open is how to choose the images used to build the LDIs in order to minimize the exposure errors that persist. Another perspective to the same problem is to try to minimize the number of images used such that building the LDIs can be done interactively.

6 CONCLUSIONS

In this paper we presented improvements to image-based rendering techniques, demonstrated on an architectural walkthrough system. The use of LDIs eliminated almost all exposure errors. Storing and warping LDIs does not cost much more than storing or warping a regular depth image. We also presented a very efficient reference-image-space clipping scheme that worked in the context of portals and architectural walkthroughs but has potential for other warping applications. Then we pointed out some precautions that need to be taken when an image is warped in occlusion compatible ordering. We also presented a correct method for load-balanced parallel warping, easily scalable to a variable number of processors.

The use of 3D image warping, in conjunction with conventional rendering of geometric models, makes for a very powerful combined system. Important objects close to the viewer can be rendered using geometry, thus preserving accuracy. More distant objects can be rendered using image-based methods, which require work proportional to image pixels not to the amount of geometry.

7 ACKNOWLEDGEMENTS

This research was primarily supported by grant number MIP-9612643 from the National Science Foundation and by the Defense Advanced Research Projects Agency under Order No. E278, Order No. A410 and DABT63-93-C-0048.

Additional support was provided by CNPq/Brazil under process number 200054/95 and a UNC Dissertation Fellowship.

We would also like to thank the UNC Walkthrough Group (for providing us with the architectural model), Matthew Rafferty and members of the UNC Graphics lab.

8 REFERENCES

[Airey90] John Airey. Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision. *Ph.D. Dissertation*, *University of North Carolina* (*also UNC Computer Science Technical Report TR90-027*), 1990.

[Aliaga97] Daniel G. Aliaga and Anselmo Lastra. Architectural Walkthroughs Using Portal Textures. In *Proceedings of IEEE Visualization '97*, pages 355-362, Oct 19-24, 1997.

[Gortler97] Steven J. Gortler, Li-wei He, Michael F. Cohen. Rendering Layered Depth Images. *Technical Report, MSTR-TR-97-09*. http://www.research.microsoft.com/pub/tr/tr-97-09.ps

[Max95] Nelson Max, Keiichi Ohsaki. Rendering Trees from Precomputed Z-Buffer Views. In Patrick M. Hanrahan and Werner Purgathofer, editors, *Rendering Techniques '95*: *Proceedings of the 6th Eurographics Workshop on Rendering*, pages 45-54, Dublin, Ireland, June 1995.

[McMillan95A] Leonard McMillan and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System. *In Proceedings of SIGGRAPH '95*, pages 39-46, August 6-11, 1995.

[McMillan95B] Leonard McMillan and Gary Bishop. Computing visibility without depth. *Computer Science Technical Report TR95-047* (October), UNC-Chapel Hill. ftp://ftp.cs.unc.edu/pub/publications/techreports/95-047.ps.Z

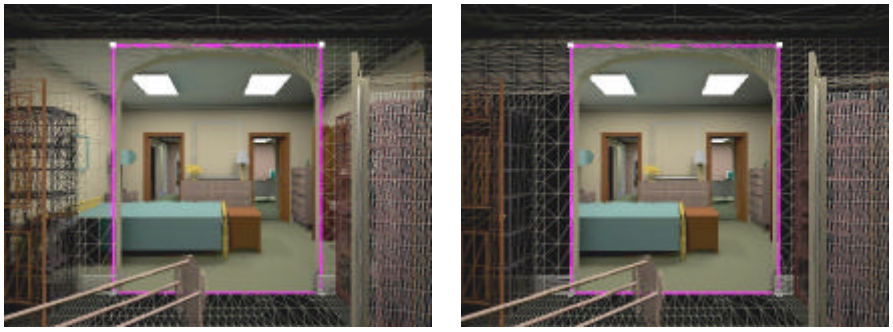
[McMillan97] Leonard McMillan. An Image-Based Approach to Three-Dimensional Computer Graphics. *Ph.D. Dissertation*, University of North Carolina, April 1997 ftp://ftp.cs.unc.edu/pub/publications/techreports/97-013.pdf.Z

[Rafferty98] Matthew M. Rafferty, Daniel G. Aliaga, Anselmo A. Lastra. 3D Image Warping in Architectural Walkthroughs. *Proceedings of VRAIS '98*, pages 228-233, March 14-18 1998.

Plate 1. *a) Artifacts when only a single image is warped. b) A layered depth image (LDI) does not exhibit the artifacts because it contains surface samples at multiple depths. c) Scene rendered with geometry for comparison.*

Plate 2*. The first two layers of the LDI. Pixels without values are shown in red.*

Plate 3. *Savings due to recursive clipping of the LDI before warping.*



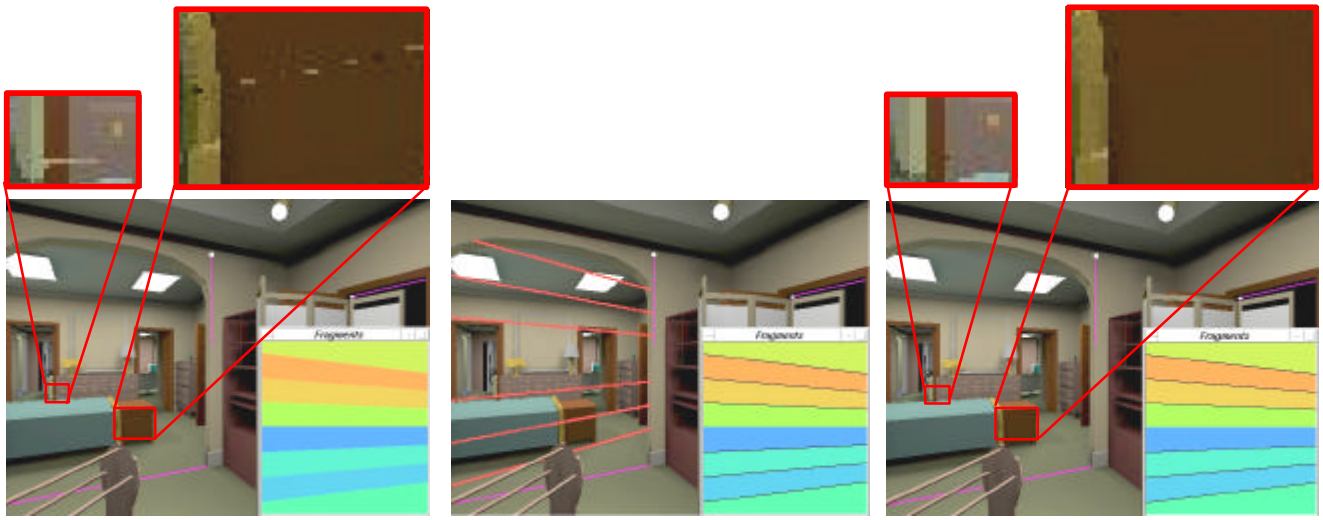


Plate 4*. Images are rendered in parallel, one fragment per processor. a) Artifacts (highlighted in red) appear at the borders between fragments because of incorrect occlusion compatible traversal. b) Buffer zones between fragments are rendered during a second pass. c) Image rendered using two-pass traversal does not exhibit the artifacts of image (a).*