

Image Based Rendering With Stable Frame Rates

Huamin Qu Ming Wan Jiafa Qin Arie Kaufman*

Center for Visual Computing (CVC) and Department of Computer Science
State University of New York at Stony Brook

Abstract

This paper presents an efficient keyframeless image-based rendering technique. An intermediate image is used to exploit the coherences among neighboring frames. The pixels in the intermediate image are first rendered by a ray-casting method and then warped to the intermediate image at the current viewpoint and view direction. We use an offset buffer to record the precise positions of these pixels in the intermediate image. Every frame is generated in three steps: warping the intermediate image onto the frame, filling in holes, and selectively rendering a group of "old" pixels. By dynamically adjusting the number of those "old" pixels in the last step, the workload at every frame can be balanced. The pixels generated by the last two steps make contributions to the new intermediate image. Unlike occasional keyframes in conventional image-based rendering which need to be totally re-rendered, intermediate images only need to be partially updated at every frame. In this way, we guarantee more stable frame rates and more uniform image qualities. The intermediate image can be warped efficiently by a modified incremental 3D warp algorithm. As a specific application, we demonstrate our technique with a voxel-based terrain rendering system.

Keywords: Image-based rendering, ray casting, voxel-based modeling, terrain rendering.

1 Introduction

Real time rendering in a walk-through or fly-through system has been a challenge in the computer graphics field for decades. The dataset of the scene model is often so large that even the most advanced rendering hardware cannot provide interactive rates. Various methods have been used to accelerate the rendering. A great deal of previous work has focused on visibility culling and level-of-detail management.

More recently, image-based rendering (IBR) techniques are used in walk-through or fly-through systems. IBR techniques exploit the frame-to-frame coherence by reusing the previously rendered images. This generally involves keyframes (or reference frames) and warping processes. Key frames are usually rendered by conventional rendering techniques from the original dataset and have

high quality. The derived frames can be generated by warping the keyframes and filling in holes if needed. Key frames can be either generated in a preprocessing phase or rendered on the fly. However, in some applications, keyframes can only be rendered on the fly. For example, in a fly-through system, users fly over a large terrain and the fly routes change interactively. It is almost impractical to pre-generate all keyframes and use them because of the huge number of keyframes needed. Thus, for such applications, keyframes are generally rendered on the fly.

However, there are some disadvantages to this keyframe method. Generally, the time to render a keyframe is higher than the time to generate a derived frame. This causes the unstable frame rates between keyframes and derived frames. The user can feel the "pause" or dramatic change in the frame rate during the fly-through. Another problem of the keyframe method is the nonuniform image quality among frames. Key frames have better image qualities than derived frames. The derived images generated at the viewpoint near the keyframes' viewpoint have better qualities than the images generated at the viewpoint far from the keyframes' viewpoint.

Most of the previous work uses this keyframe IBR technique on surface models. When we try to use this keyframe technique in a voxel-based scene model, problems become more serious. The conventional rendering technique in a voxel-based scene model is ray casting. The ray-casting method can generate a high quality image, but it is time-consuming. Thus, the cost to generate a keyframe on the fly is high, and the unstable frame rate problem becomes more serious.

One advantage of the ray-casting rendering method is that it is easy to render any pixel in an image by casting one ray. The holes in the derived frames can be filled in by the ray-casting method. Actually, the ray-casting method can easily render any part, even scattered pixels of an image. We try to take advantage of this feature in this work. The basic idea is to use an image warping technique to gain a high frame rate, and to use the ray-casting method to selectively re-render part of the image to guarantee image quality.

We present an efficient image-based rendering technique without keyframes in the context of voxel-based scene modeling. We use an intermediate image to exploit the coherence in frames. The pixels in the intermediate image are all first generated by the ray-casting method in previous frames and then warped to the intermediate image at the current viewpoint and view direction. We relax the requirement that all pixels in an image are located in a regular rectangular grid. We use an offset buffer to record the precise position of these pixels in the intermediate image. Every frame is generated in three steps: warping the intermediate image, filling in holes and selectively rendering some other pixels. This last step can guarantee the image quality. By dynamically adjusting the number of those pixels in the last step, we can balance the workload at every frame. The pixels generated by the last two steps make contributions to the new intermediate image.

McMillan and Bishop's 3D warp algorithm [10, 11] can be successfully adapted to warp the intermediate image. We show that the intermediate image can still be warped in McMillan's occlusion compatible order [9]. By quantizing the offset of pixels in the intermediate image and using lookup tables, the advantage of the

*Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA. Email: {huamin|mwan|jiafa|ari}@cs.sunysb.edu

incremental computation of McMillan and Bishop’s algorithm can be preserved.

The advantage of our method over the keyframe method is that the keyframe needs to be rerendered totally, but the intermediate image only needs to be updated partially at every frame. By combining the advantages of the image-based rendering techniques and the ray-casting method, we can get more stable frame rates and more uniform image qualities. Our method is especially useful in a real time system when the keyframe rendering is time consuming. We demonstrate our method with a voxel-based terrain rendering system.

This paper is organized as follows. Section 2 reviews the previous work in image-based rendering techniques and voxel-based terrain modeling. Our keyframeless image-based rendering technique is described in detail in Section 3, which focuses on the construction and warping of the intermediate image. Section 4 presents results from our implementation. Conclusions and future work are discussed in Section 5.

2 Previous Work

2.1 Image-based rendering

In recent years, there have been many papers on image-based rendering. Mark’s [8] is a good survey. Some researchers have used the keyframe method to accelerate rendering in a walk-through or fly-through system.

Mark [7] has warped two different reference images and composited the results to avoid occlusion-related artifacts. The reference images are generated on the fly based on the prediction of the future viewpoint and view direction. The reference frames are rendered at 5 frame/sec, and derived frames can be generated at 30 frame/sec.

Popescu et al.[12] have warped layered depth images in the context of an architectural walk-through system. The layered depth images are synthesized in a preprocessing phase for every portal.

Chen et al.[2] used a hybrid LOD-Sprite technique to accelerate terrain rendering. A sprite image is generated from high-resolution scene geometry at every keyframe. Then the sprites are reused by texture mapping in the following frame using low-resolution scene geometry. The keyframes are generated on the fly. However, the time to render a keyframe can be as high as three times more than the time to render a derived frame.

Instead of using an image to represent the entire scene, the impostor methods [6, 14] uses image to represent remote objects in a scene.

Among various warping algorithms, McMillan and Bishop’s 3D warp method [10, 11] is an efficient algorithm. It warps the image with depth information, so it can provide proper parallax and can be implemented efficiently by incremental computation of the 3D warp equation and by using an occlusion compatible ordering algorithm to resolve occlusion without z-buffering. McMillan and Bishop’s algorithm relies on the epipolar geometry. Given two images, the epipole in the second image is the projection of the camera location of the first image into the second image. The epipolar lines are the image-space lines that pass through the epipole. The relative warp order for different epipolar lines does not matter, but the points on a single epipolar line must be warped in a particular order. Thus, the input image is split into sheets horizontally and vertically at the epipolar point. Every sheet is processed in a different scan line order. Popescu et al.[12] pointed out that special attentions should be paid to the pixels on the row and column of the epipole because pixels in discrete images have a non-negligible area. They have to be warped either first or last, depending on the sign of the epipole.

Our work is based on McMillan and Bishop’s algorithm.

2.2 Voxel-based terrain modeling

Most of the current flight simulators are based on the surface model. However the voxel-based approach has many advantages compared to a surface-based one. For example, the format of the elevation map lends itself to generating a very high resolution 3D volume of terrain and multi-resolution volumes. Also, texture mapping for voxels is much simpler, of higher quality, and can be preprocessed. More importantly, the voxel-based model is somewhat scene complexity independent, and it is easy to incorporate clouds, haze, flames, and other amorphous phenomena and volumetric objects [4]. Therefore, in recent years, the voxel-based approach has become a new research issue for height-field visualization systems [18, 5, 3].

Most of the published voxel-based terrain rendering algorithms directly use the 2D elevation map to represent 3D values, so that both memory size and accessing time for the terrain model can be greatly reduced [5, 3]. However, the quality of images generated from this model is not very good. Thus, a true 3D voxel-based terrain model has been generated in our previous voxel-based terrain visualization system [16]. The terrain model consists of a 3D volumetric terrain dataset and a corresponding aerial photograph or satellite image of the terrain. There are also some disadvantages of this 3D voxel-based terrain model. The storage requirement for a 3D voxel-based terrain is high, but memories are becoming larger and relatively inexpensive. If there is no hardware support, it usually takes longer time to render a 3D voxel-based terrain by ray-casting than to render a surface-based one with the hardware support for texture mapping.

3 3D Warp Without Keyframe

3.1 Keyframeless rendering

In a walk-through or fly-through system, the viewpoint changes gradually. There is substantial coherence in adjacent frames. A straightforward way to take advantage of this is to render some keyframes by the ray-casting method and then warp the keyframes to get the next frames. When the error is beyond a threshold, the keyframe is rerendered. Because of the change of viewpoint, some new part of the scene enters the user’s view and some occluded part of the scene in the keyframe can pop up in the new frame. This causes holes in the resulting image. These holes can be filled in by ray casting. Figure 1 shows this method.

As we mentioned before, this keyframe method has the disadvantages of unstable frame rates and nonuniform image qualities. This method only takes advantage of the coherence between the keyframe and the derived frame. There are also coherences in the derived frames. Let’s take a look at Figure 1. Frame 1 is the keyframe generated by using the ray-casting method. Frame 2 to Frame n are derived frames by warping Frame 1. Actually, at every derived frame, except the pixels warped from the keyframe, a certain amount of pixels have to be rendered by the ray-casting method because of the holes in the warped images. These pixels have a high image quality and likely will appear in the next frames because of the coherence in the adjacent derived frames. To make a difference, we use the term “high quality pixels” to refer to the pixels generated by the ray-casting method in a derived frame and “low quality pixels” to refer to pixels reconstructed from the warped image. Traditionally, the keyframe method does not take advantage of this.

There are two intuitive methods that can take advantage of coherences in derived frames:

- 1) Warping frame m-1 to get frame m. Image quality degrades

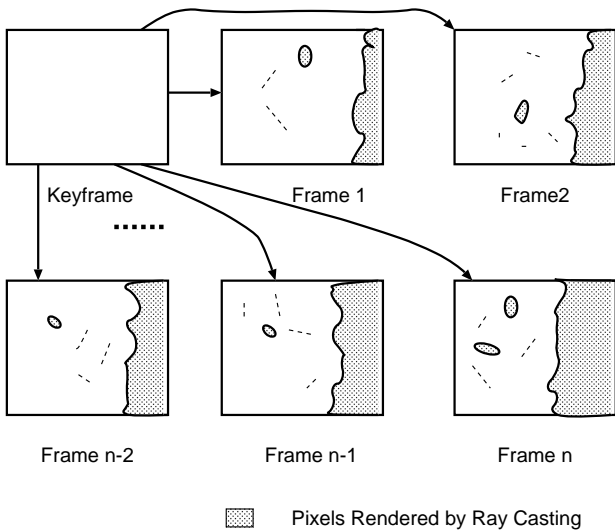


Figure 1: A straightforward keyframe technique. The keyframe is rendered by the ray-casting method. Frames 1-n are rendered by warping the keyframe. The holes in frames 1-n are filled in by ray casting. When the error is beyond a threshold, the keyframe is re-rendered.

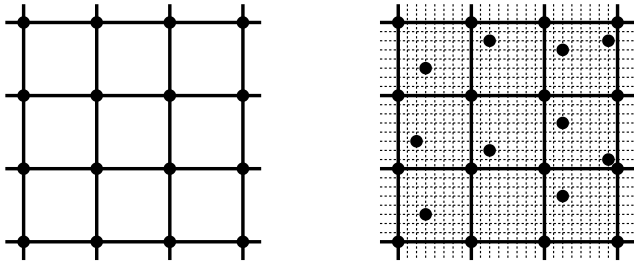


Figure 2: Offset buffer. The exact positions of the warped pixels are recorded by an offset buffer.

quickly after several such warps because most pixels in frame $m-1$ are from reconstruction and have low image quality.

2) Warping the high quality pixels from the previous $m-1$ frames to get frame m . Then, the Z-buffer has to be used to resolve the occlusion. We have to warp scattered points in $m-1$ frames. The advantage of the incremental computation does not exist.

We try to design an image-based rendering technique which exploits the coherences in the multi-frames which meet the following requirements :

- 1) Only high quality pixels can be warped. We do not warp the pixels from reconstruction.
- 2) No Z-buffer is needed to resolve the occlusion problem
- 3) Preserve the advantage of incremental computation.
- 4) No keyframe is needed.

As shown in Figure 2, when a pixel is warped from the keyframe to frame 1, the center of the pixel generally does not fall exactly on the grid of the derived frame. Thus a reconstruction process is needed. However, if we store the exact position of the center of this pixel in frame 1, warping this pixel from frame 1 to frame 2 will have the same effect as warping the original pixel from the keyframe to frame 2.

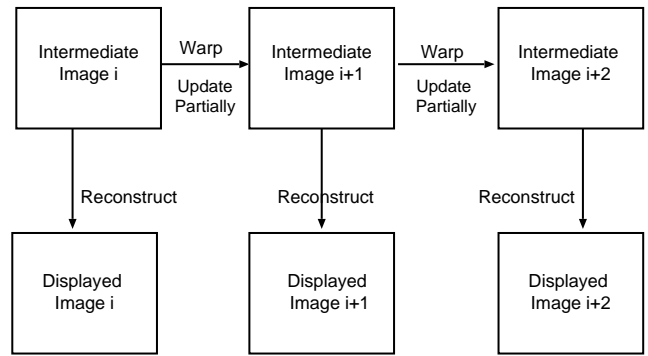


Figure 3: The keyframeless IBR method. The intermediate image $i+1$ is rendered by warping the intermediate image i . The exact positions of warped pixels in the intermediate image $i+1$ are recorded by the offset buffer. After that, the intermediate image $i+1$ is partially updated by ray casting according to the ages of the pixels. The displayed image $i+1$ is reconstructed from the intermediate image $i+1$. The first intermediate image is rendered by ray casting.

We refer to the final reconstructed image as the displayed image, and refer to the warped image before reconstruction as the intermediate image. The exact positions of warped pixels in the intermediate image are recorded by an offset buffer which represents the offset of the center of the pixels from their nearest neighborhood. The pixels in the intermediate image are all first rendered by ray casting in a previous frame and then warped into the intermediate image at the current viewpoint and view direction.

At the beginning, the first frame is generated by the ray-casting method. Thus, the intermediate image is exactly the same as the displayed image. The offsets of all pixels at the intermediate image are 0. Then, we warp the intermediate image to get the next intermediate image. The offsets of warped pixels are recorded by an offset buffer. Now there are some holes in the intermediate image. We fill in these holes by the ray-casting method. The displayed image is reconstructed from this intermediate image. Then, we warp this intermediate image to get the next intermediate image and reconstruct the next displayed image from the next intermediate image, and so on.

Therefore, at any intermediate image some pixels are newly rendered by the ray-casting method. Some pixels are warped from the previous intermediate image. In order to make a difference between these two types of pixels, we use an age buffer to record the age of a pixel. If a pixel is just rendered in this image by the ray-casting method, its age is 0. If a pixel is warped from the previous intermediate image and is still visible at the current intermediate image, then its age is incremented by 1. By using the age buffer, we know exactly how many times a pixel has been warped from the intermediate image where this pixel is first generated by the ray-casting method to the current intermediate image. The older the pixel is and the longer the pixel stays in the intermediate image, the more likely it will disappear in the next frame, and more important, the more likely the image quality around this pixel will be degraded. Thus, the age of a pixel is a nice criteria for the image quality around this pixel.

In order to guarantee the image quality, we set a threshold for the ages of the pixels in the intermediate image. If the age of a pixel is older than this threshold, then we do not warp this pixel. This probably causes holes in the next intermediate image and these holes are filled by the ray-casting method.

Therefore, the pixels in the intermediate image are all high qual-

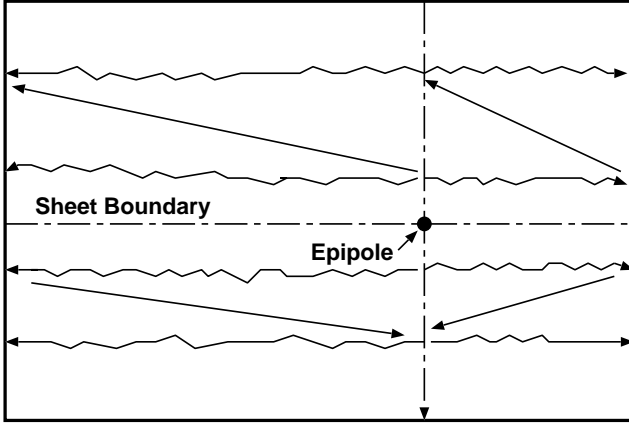


Figure 4: Warp order of the intermediate image for a negative epipole.

ity pixels. By using the offset buffer to record the exact positions of these pixels, warping these pixels from the intermediate image has the same effect as warping the original pixels from the image where these pixels were first generated by the ray-casting method. By using the age buffer, we can control the image quality by setting a threshold for ages of pixels in the intermediate image. Figure 3 shows our keyframeless IBR method.

Please note that Popescu et al. [13] have used the offset buffer for a different purpose. They used the offset buffer to resolve the visibility of the pixels. We use the offset buffer to record the precise positions of the pixels in the intermediate image for further warping.

3.2 Warping the intermediate image

The pixels in the intermediate image no longer fall in the regular rectangular grid. However, as shown in Figure 4, the intermediate image can still be warped by McMillan and Bishop's ordering algorithm. As we mentioned before, there are two properties of the occlusion-compatible order [8]: (1) the relative warp order for two reference-image points on different reference-image epipolar line does not matter; (2) the points on a single reference-image epipolar line must be warped in a particular order. Even though the scan line in the intermediate image is no longer a straight line, these two properties are still preserved. The formal proof of McMillan [11] applies, so the ordering algorithm still works.

One advantage of McMillan and Bishop's 3D warp algorithm is the incremental computation. We use the notation of the pinhole camera model adopted by McMillan [11]. Then the 3D warp equation is [8]:

$$\frac{z_2}{S_2} \frac{S_1}{z_1} \bar{u}_2 = P_2^{-1} P_1 \bar{u}_1 + P_2^{-1} \frac{S_1}{z_1} (\hat{C}_1 - \hat{C}_2)$$

$$S = \frac{\vec{a} \times \vec{b}}{\|\vec{a} \times \vec{b}\|} \cdot \vec{c}, \bar{u}_1 = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}, \bar{u}_2 = \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}.$$

The vectors $\vec{a}, \vec{b}, \vec{c}$ are the basis vectors for the camera coordinate system. P and \hat{C} represent the pinhole camera viewing parameter and center of projection respectively for the images. The \bar{u}_1, \bar{u}_2 are

the image coordinates of the reference frame and the derived frame. z_1 and z_2 are the reference frame and derived frame depth values.

To compute the warped position of the next pixel along a scan line, incremental computation can be used:

$$P_2^{-1} P_1 \cdot \begin{bmatrix} u_1 + 1 \\ v_1 \\ 1 \end{bmatrix} = P_2^{-1} P_1 \cdot \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} + P_2^{-1} P_1 \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= start + uincr$$

For an intermediate image, the next pixel coordinates at the image space along a scan line is $(u + 1 + offset_u, v + offset_v, 1)$ instead of $(u + 1, v, 1)$. The incremental computation cannot be directly used.

As mentioned before, we set a threshold for the ages of the pixels in the intermediate images. That means the number of times a pixel can be warped is bounded by a threshold. We can use a fixed point instead of a float point to improve the efficiency. We can quantize the offset of pixels into levels. Let's suppose at image space the one pixel distance is quantized into L levels. At every frame, the warped pixel position error introduced by the quantization is $0.5 \times (1/L)$. Thus, the total possible error introduced by the quantization after m warps is bounded by $c \times m \times (1/L)$, with c as a constant.

Because the offset is quantized, we can precompute $P_2^{-1} P_1 \cdot [offset_u, 0, 0]$ and $P_2^{-1} P_1 \cdot [0, offset_v, 0]$ for all possible levels of the offset and store them in two lookup tables $table_u$ and $table_v$. Before rendering each new image, we use the new camera information to precompute values for all lookup table indices. Then, McMillan and Bishop's incremental computation method can be adapted by adding two items to compensate for the offset of the pixels from the regular grid:

$$P_2^{-1} P_1 \cdot \begin{bmatrix} u_1 + 1 + offset_u \\ v_1 + offset_v \\ 1 \end{bmatrix} =$$

$$P_2^{-1} P_1 \cdot \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} + P_2^{-1} P_1 \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} +$$

$$P_2^{-1} P_1 \cdot \begin{bmatrix} offset_u \\ 0 \\ 0 \end{bmatrix} + P_2^{-1} P_1 \cdot \begin{bmatrix} 0 \\ offset_v \\ 0 \end{bmatrix}$$

$$= start + uincr + table_u[offset_u] + table_v[offset_v]$$

3.3 Filling in holes

The holes in the intermediate image are filled in by the ray-casting method. It can be done efficiently by exploiting the coherence between rays. For example, we show how to fill in holes in a voxel-based terrain rendering system. We fill in the holes in the warped image column by column. In each column, we cast one ray through each hole pixel bottom up, in order to speed up ray traversal, by exploiting the specific vertical ray coherence [5, 3] in terrain scenes. As shown in Figure 5, the basic idea of ray coherence is that for two viewing rays from the same camera that project onto the same line on the base plane of the terrain, the higher ray hits the terrain surface at a position farther away from the viewpoint. By exploiting this ray coherence, ray casting can be dramatically accelerated by skipping most of the empty space above the terrain surface.

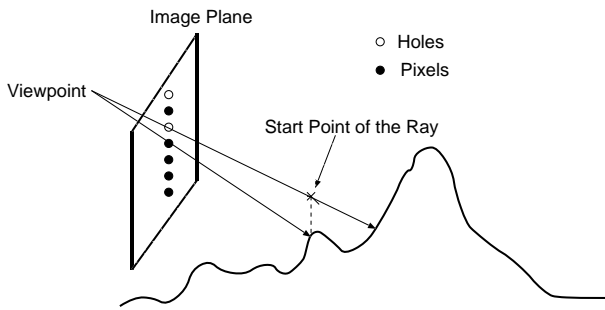


Figure 5: The ray to fill in a hole can emanate from the hit point of the pixel just below it

During our hole filling procedure, ray casting for each image column is completed in the following steps: first, find the location of the lowest hole pixel and the depth of the pixel just below this pixel from the current intermediate image buffer. Cast a ray through this pixel and use the corresponding depth to skip the empty space along the ray. In a special case when the lowest hole pixel is at the bottom of the column, we have to traverse the ray from the image without any space leaping. Once the hit position is found along the ray, save the new depth value. Second, move upward along the column to the next hole pixel and cast a ray through it. If its lower pixel is a ray-casting pixel, we use its lower pixel depth for space leaping; otherwise, its lower pixel is a warped pixel, and we use the lower pixel depth from warping for space leaping. Third, save the new depth once the hit point is found. Fourth, repeat the second step, until no hole pixels are left or the current ray does not intersect with the terrain surface. Finally, search upward along the column for all remaining ray-casting pixels, and directly assign their colors to be background color. No ray casting is performed through these pixels, since they have no chance of intersecting with the terrain surface.

3.4 Balancing the workload

At every viewpoint, we need to warp the intermediate image at the previous viewpoint and fill in the holes. The time to warp an image is independent of the scene complexity and almost a constant. The time to fill in the holes by ray casting depends on the number of holes, which can vary from frame to frame. As we mentioned before, the holes are caused by :

- 1) Some region of terrain just enters the current view.
- 2) Some region of terrain occluded in the previous frame now pops up.
- 3) The ages of some pixels at the intermediate image are older than a threshold, so we do not warp these pixels.

The total number of these pixels should be controlled by our computational ability. We also hope that the numbers of pixels needed to be re-rendered by ray casting at every frame are almost the same. We cannot control the number of holes caused by (1) (2). However, we can control the number of holes caused by (3) by adjusting the threshold. The way to balance the workload at every frame is to dynamically adjust the number of pixels caused by (3).

We set a threshold for the total number of pixels we can afford to re-render by ray casting. Then, at every frame, after warping the intermediate image, we count the number of holes. If the number is beyond our threshold, we can do nothing else except fill in these holes. However, if the number is below our threshold, we compute the difference of the threshold and the number of our current holes.



Figure 6: Camera Path.

Then we can select that number of pixels from the pixels which are the oldest in the intermediate image and re-render them by ray casting. In this way we can balance the workload and guarantee the image quality at the same time.

3.5 Reconstruction

Reconstruction is a difficult problem in image-based rendering techniques. Chang et al.[1] use a bilinear kernel. Four LDI pixels are updated for each pixel of a reference image. Shade et al.[15] use a rough approximation to the footprint evaluation optimized for speed. The image size is approximated by using a lookup table. The four splat sizes they used have 1×1 , 3×3 , 5×5 , and 7×7 pixel footprints. The alpha values are rounded to 1, 1/2, or 1/4. Therefore, the alpha blending can be done with integer shifts and adds. Mark et al.[7] use a technique to treat the reference frame as a mesh. The 3D warp perturbs the vertices of the mesh. Reconstruction occurs by rendering the perturbed mesh triangles into the derived frame.

For simplicity, we use one pixel-wide reconstruction kernel. We just write a single pixel in the intermediate image in the nearest neighbor position of the derived image.

4 Experimental Results

Our method was implemented on a Silicon Graphics *Onyx²* (1024MB RAM, four R1000 processors) with Infinite Reality graphics. However, we did not exploit its parallel processing capability in our implementation. Only one processor was used.

We demonstrated our method with a voxel-based rendering system [16]. Our terrain model consists of a 3D terrain volume with a resolution of $512 \times 512 \times 64$ and a corresponding registered aerial photo. Figures 7 give the experiment results of our algorithm for a camera path shown at Figure 6 which produces more than 300

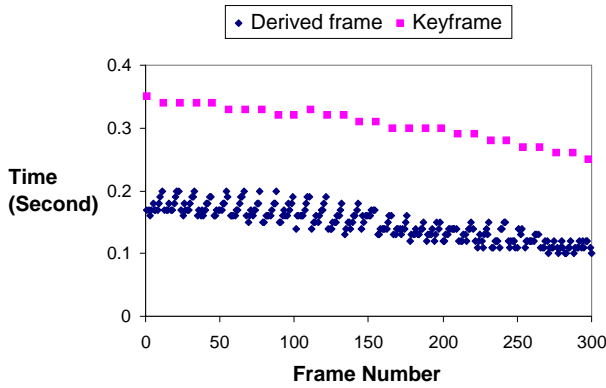


Figure 7: Time to render each frame by the keyframe IBR method.

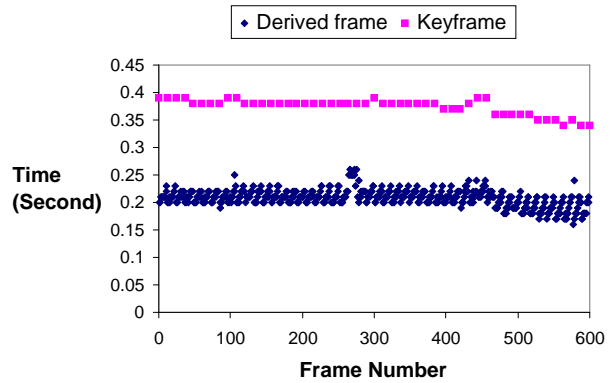


Figure 9: Time to render each frame by the keyframe IBR method.

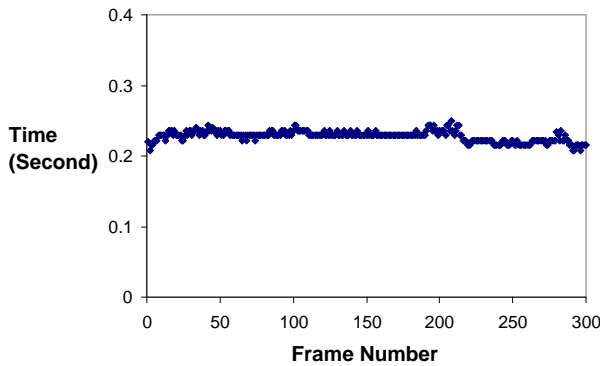


Figure 8: Time to render each frame by our keyframeless IBR method.

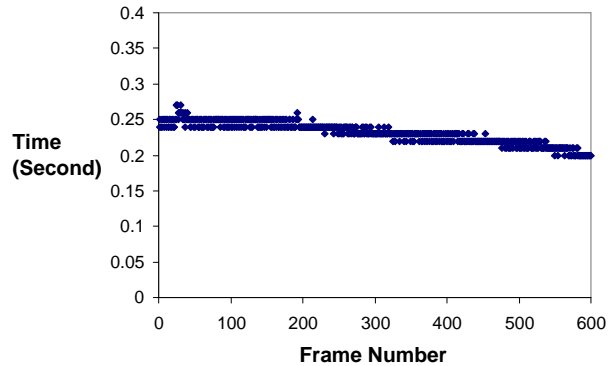


Figure 10: Time to render each frame by our keyframeless IBR method.

frame images. The image size is 500×400 . The ray-casting rendering method used in this paper is based on our earlier work[16].

Figures 7 and 8 show the amount of time required to render each frame. We compared our algorithm with the keyframe algorithm. Figure 7 shows the result of the keyframe method. We rendered the keyframe by the ray-casting method on the voxel terrain model. Ten derived frames were gotten by warping one keyframe. We can see that the time to render keyframes is about two times longer than the time to render the derived frames. Even the time to render derived frames can be quite different. Figure 8 shows the result of our method. The threshold for the age of pixels is 10. The number of pixels needed to be rerendered is about 14 percent of the total number of pixels in the intermediate image. Comparing Figure 7 to Figure 8, we can see that our method can get a more stable frame rate than the keyframe method.

Figures 11 and 12 show a view of the terrain (Frame No. 100). Figure 11 shows the terrain rendered by the ray-casting method. Figure 12 shows the terrain rendered by our keyframeless IBR method. Figure 13 gives the absolute value of the difference between these two images. We quantized the offset of pixels into 1024 levels. Because the threshold for the age of pixels is 10, we think that the position errors of pixels in the intermediate image caused by quantization and at most 10 consecutive warpings are negligible. The image quality of our algorithm depends on the reconstruction method. We used one pixel-wide reconstruction kernel and got sat-

isfying results. If more precise reconstruction methods are used the image quality can be further improved.

We also demonstrated our method on another terrain. The terrain size is $468 \times 693 \times 64$. Figures 9 and 10 show the amount of time required to render each frame. Figure 9 shows the result of the keyframe method. Figure 10 shows the result of our method. Figure 14 shows the terrain rendered by the ray-casting method. Figure 15 shows the terrain rendered by our keyframeless IBR method. Figure 16 shows the difference between these two images. From these figures we got the following observations: our method get a more stable frame rate than the keyframe method. The quality of the images generated by our method is comparable to that of ray casting and can be sustained at that level.

5 Conclusions and Future Work

We have presented a novel image-based rendering technique used in voxel-based scene modeling. Our contributions are:

- 1) We designed a keyframeless image-based rendering technique which combines the advantages of the 3D warp algorithm and the ray-casting method. The 3D image warp algorithm is used to gain a high frame rate, and the ray-casting method is used to fill in holes and to selectively rerender part of the image to guarantee the image quality.

2) An offset buffer is used to record the precise position of a pixel, so pixels can be consecutively warped from a single intermediate image without loss of accuracy. McMillan and Bishop's warp algorithm is adapted to warp the intermediate image. By quantizing the offset, we can preserve the incremental computation.

3) An age buffer is used to record the age of a pixel. This provides a nice criteria for image quality. By slightly adjusting the threshold for the ages of pixels in the intermediate image, we can balance the workload at every frame. Our method indeed obtains more stable frame rates and more uniform image qualities.

We demonstrate our method with a voxel-based terrain rendering system. However, our method can be applied to many other applications. It is especially useful in the applications which have the following features :

- 1) The keyframes have to be rendered on the fly and the time to render one keyframe is much higher than the time to generate one derived frame.
- 2) Ray casting can be used as a rendering method.

There are also some limitations to our method. First, the irregular distribution of pixels in the intermediate image can cause extra difficulty for reconstruction. If the nearest neighbor is used as the reconstruction method, we found that warping the intermediate image can cause more holes than warping the keyframe. Second, like other image-based rendering methods the performance of our method depends on the frame-to-frame coherence. If there is nice frame-to-frame coherence, such as in the terrain dataset, our method can give pretty stable frame rates and uniform image qualities. If the frame-to-frame coherence is not that good, such as in a dataset with a very complicated background scenery, then it is possible that the number of holes per frame can be vary substantially, and the ray casting may take longer than expected to fill in these holes. This may lead to inconsistent frame rates.

Some work need to be done in the future. We want to find a more accurate and fast reconstruction method for our intermediate image. We plan to use a splatting method instead of the current nearest neighbor method, so the number of holes can be decreased and the image quality can be improved. We also plan to parallelize our method to achieve higher frame rates.

Acknowledgments

This work is partially supported by ONR grant N000149710402. Thanks to Nan Zhang, Baoquan Chen, and other members of the virtual fly-through project. Thanks to Manuel Menezes de Oliveira Neto and anonymous reviewers for their valuable comments. We would also like to thank Marianne Catalano for proofreading.

References

- [1] Chun-Fa Chang, Gary Bishop, Anselmo Lastra. LDI Tree: A Hierarchical Representation for Image-based Rendering. *SIGGRAPH 99 Conference Proceedings* , pages 291-298, August 1999.
- [2] Baoquan Chen, J. Edward Swan II, Eddy Kuo, Arie Kaufman. A Hybrid LOD-Sprite Technique for Accelerated Terrain Rendering. *Proceedings of IEEE Visualization '99* , pages 291-298, October 1999
- [3] D. Cohen-Or, E. Rich, U. Lerner, and V. Shenkar. A Real-Time Photo-Realistic Visual Fly-through. *IEEE Transactions on Visualization and Computer Graphics* , 2(3), pages 255-265, 1996.

- [4] A. Kaufman, D. Cohen and R. Yagel. Volume Graphics. *Computer* , 26(7), pages 51-64, 1993.
- [5] C. Lee and Y. Shin. An Efficient Ray Tracing Method for Terrain Rendering. *Pacific Graphics '95* , pages 181-193, 1995.
- [6] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. *Proceedings of the 1995 Symposium on Interactive 3D Graphics* , pages 95-102, April 1995.
- [7] William R. Mark, Leonard McMillan and Gary Bishop. Post-Rendering 3D Warping. *Proceedings of the 1997 Symposium on Interactive 3D Graphics* , pages 7-16, April 1997.
- [8] William R. Mark. Post-Rendering 3D Image Warping: Visibility, Reconstruction, and Performance for Depth-Image Warping. Ph.D. Dissertation, University of North Carolina, April 1999. (Also UNC Computer Science Technical Report TR99-022).
- [9] Leonard McMillan and Gary Bishop. Head-tracked stereoscopic display using image warping. *Proceedings SPIE* , volume 2409, pages 21-30, February 1995.
- [10] Leonard McMillan and Gary Bishop. Plenoptic Modeling. *SIGGRAPH 95 Conference Proceedings* , pages 39-46, August 1995.
- [11] Leonard McMillan. An Image-Based Approach to Three-Dimensional Computer Graphics. Ph.D. Dissertation. Technical Report 97-013, University of North Carolina at Chapel Hill. April 1997.
- [12] Voicu S. Popescu, Anselmo A. Lastra, Daniel G. Aliaga, Manuel de Oliveira Neto. Efficient Warping for Architectural Walkthroughs using Layered Depth Images. *IEEE Visualization '98* , pages 211 - 215, October 1998.
- [13] Voicu S. Popescu, and Anselmo Lastra. High Quality 3D Image Warping by Separating Visibility from Reconstruction. UNC Computer Science Technical Report TR99-017, University of North Carolina, April 5, 1999.
- [14] G. Schaufler. Dynamically Generated Impostors. In D. W. Fellner, editor, *Modeling Virtual-Worlds-Distributed Graphics, MVD 95 Workshop* , pages 129-136, November 1995.
- [15] Jonathan Shade, Steven Gortler, Li-wei He and Richard Szeliski. Layered Depth Images. *SIGGRAPH 98 Conference Proceedings* , pages 231-242, July 1998
- [16] Ming Wan, Huamin Qu, and Arie Kaufman. Virtual Fly-through over a Voxel-Based Terrain. *Proceedings of IEEE Virtual Reality 1999* , pages 53-60, 1999
- [17] Lee Westover. Footprint evaluation for volume rendering. *Computer Graphics (SIGGRAPH 90 Conference Proceedings)* , volume 24, pages 367-376, August 1990.
- [18] J. Wright and J. Hsieh. A Voxel-Based, Forward Projection Algorithm for Rendering Surface and Volumetric Data. *IEEE Visualization '92* , pages 340-348, 1992.

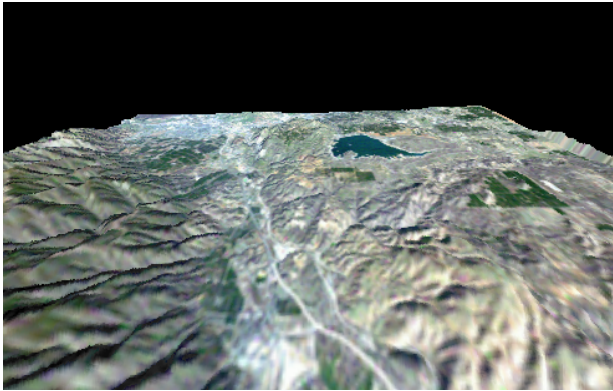


Figure 11: A California terrain (size: $512 \times 512 \times 64$) rendered by the ray-casting method. (See also Color Plate.)

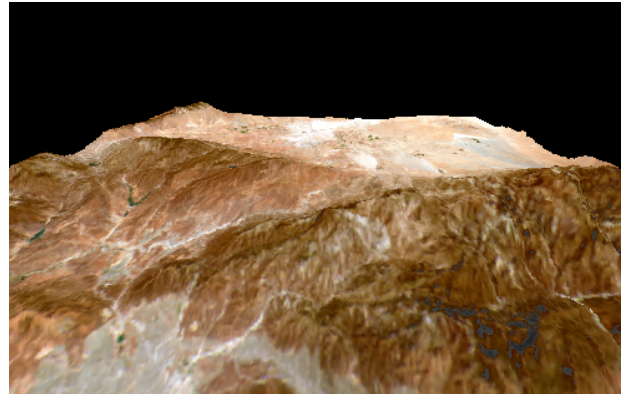


Figure 14: A Los Angeles terrain (size: $468 \times 693 \times 64$) rendered by the ray-casting method. (See also Color Plate.)

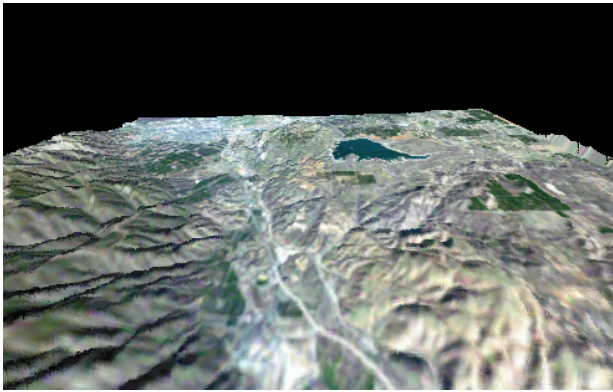


Figure 12: A California terrain (size: $512 \times 512 \times 64$) rendered by our keyframeless IBR method. (See also Color Plate.)

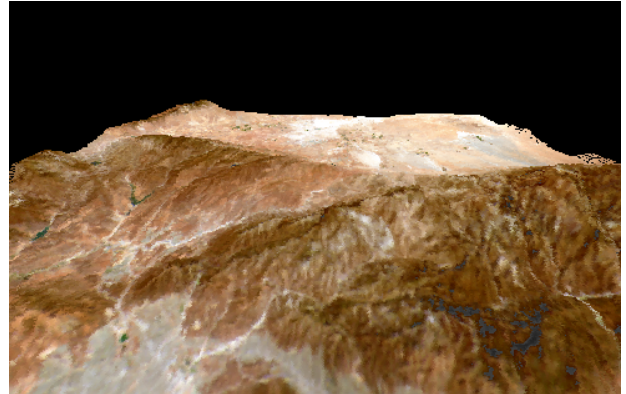


Figure 15: A Los Angeles terrain (size: $468 \times 693 \times 64$) rendered by our keyframeless IBR method. (See also Color Plate.)

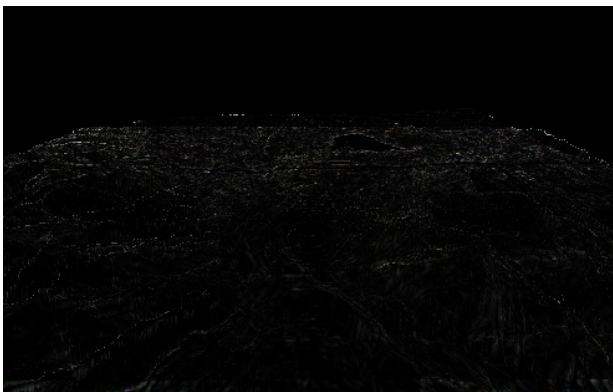


Figure 13: The difference between Figure 11 and Figure 12. (See also Color Plate.)

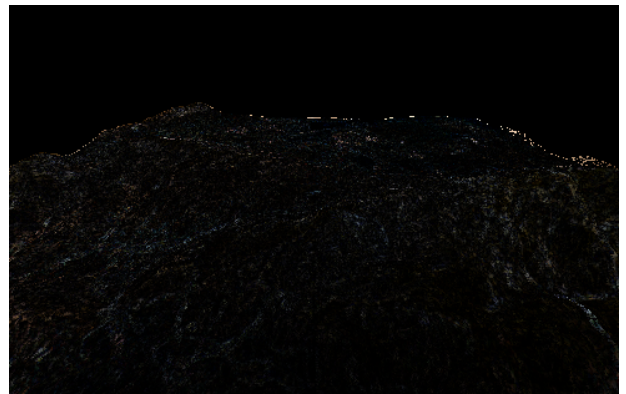


Figure 16: The difference between Figure 14 and Figure 15. (See also Color Plate.)