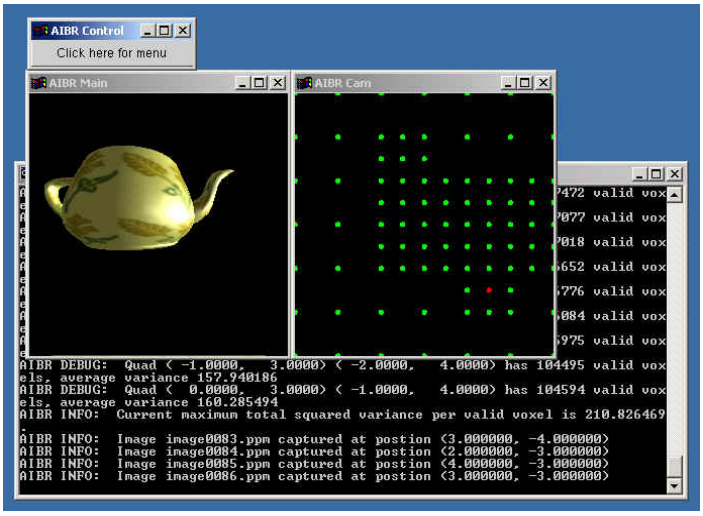
Carnegie Mellon

Electrical & Computer ENGINEERING

Active Scene Capturing for Image-Based Rendering with a Light Field Setup

Cha Zhang and Tsuhan Chen
Advanced Multimedia Processing Lab



Technical Report AMP 03-02
March 2003

Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

With image-based rendering (IBR) one can render 3D scenes from multiple images with or without geometry. Various approaches to IBR have been proposed to render the scene given a set of regularly or non-regularly pre-captured images. In this paper, we propose an algorithm to capture the images actively. Based on the images that have been taken, the system intelligently determines where to pose the next camera for best rendering performance. This results in non-uniform but optimal capturing of the scene. We also propose a local-consistency voxel coloring algorithm that can find 3D geometry for non-Lambertian scenes. The full active capturing algorithm combines the above two components and is able to render better quality images compared with traditional methods.

1. Introduction

Image-based rendering (IBR) has received much attention in the last few years. Unlike traditional rendering techniques where the 3D geometry is known, IBR can render 3D scenes directly from captured images. Geometry information can be incorporated into IBR but it is no longer the dominant factor.

Previous work has been focused on different IBR representations and rendering methods [1]. When dense image samples are available, new views can be rendered by interpolation, such as plenoptic modeling [2], light-field [3], Lumigraph [4] and concentric mosaics [5]. 3D correspondence was used in several IBR approaches, e.g., view interpolation [6], view morphing [7] and 3D modeling from images [8]. When some sort of geometry information is known, it can be incorporated into IBR as well. Representative works are 3D warping [9], layered depth image [10], view-dependent texture mapping [11] and unstructured lumigraph [12].

Although the images used during the rendering are captured in different ways, the literature has extensively focused on how to render the scene given a set of images, with or without geometry. However, little work has been performed on the capturing process, which is as important as the rendering process. Recently there has been some work on the minimum sampling requirement of IBR [13][14]. When less images than the minimum requirement are captured, ghosting effects or aliasing will take place during the rendering. Although aliasing can always be avoided by capturing more than enough images, in reality we have to constrain the number of images being captured due to storage concerns. It is thus important to know where to pose our cameras for the best overall rendering quality.

We use two criteria to measure the rendering quality in this paper. The first is the worst-case quality, i.e., the worst image quality when we move around the virtual camera during the rendering. The second measure is

the variance of the image qualities during the rendering. A large variance means that the image qualities are very unstable, which causes bad experience to the viewer. The above two criteria can be both optimized with one approach, i.e., to always capture images at places where the rendering image quality is the worst. This is the key idea of active capturing for image-based rendering (AIBR).

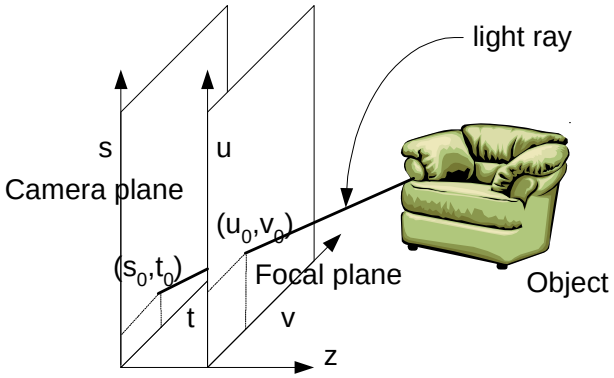


Figure 1 Lightfield parameterization.

Albeit the diversity of IBR representations, we consider a scenario similar to the lightfield [3]. As shown in Figure 1, we constrain that the cameras are on a common camera plane (indexed by (s,t)), and they share the same focal plane. While in lightfield the cameras are arranged on a regular grid, we allow the camera to be any where on the plane. From the signal processing point of view, we employ non-uniform sampling to lightfield instead of the uniform sampling in the regular lightfield. The extra freedom makes it possible to capture the scene better than the conventional approach.

We also build a voxel model for the scene. This is similar to Lumigraph [4][12], where geometry information is used to improve the rendering performance. Instead of the octree construction algorithm in [4], we use a modified voxel coloring algorithm [15] to recover the geometry, as will be shown in Section 3-C. The voxel model is also used to estimate the rendering quality for AIBR, as is shown in Section 3-A.

Our AIBR algorithm iterates between two stages. In the first stage, we fix the geometry and determine where to capture more images. This is based on the color consistency verification for the given voxel model. The second stage refines the voxel model if necessary. A modified voxel coloring algorithm based on local consistency is proposed to find the voxel model of the scene even when it is highly non-Lambertian.

The paper is organized as follows. In Section 2 we describe some previous work that is related to our approach. Section 3 presents the details of our proposed method. Experimental results are shown in Section 4. Conclusions and future work are given in Section 5.

2. Related Work

Our capturing and rendering process is very similar to the Lumigraph [4] and unstructured Lumigraph [12]. Both the previous approaches employed non-uniform sampling. In [4], a user interface was developed to display the current and previous camera position on a viewing sphere. It is the user's responsibility to determine where more images are needed. The unstructured Lumigraph [12] was rendered from video sequences captured by hand-held camera with an arbitrary camera path. Such capturing methods suffer in the rendering stage because depending on where the images were captured, the rendering image quality can be very unstable. There is no guarantee about the worst rendering quality, either.

Another related work is the voxel coloring algorithm [15]. For a Lambertian scene, voxel coloring algorithm states that a voxel belonging to the scene surface should be color invariant across all the visible images. As presented in Section 3-0, we find that such color consistency property is essential to image-based rendering, too. In Section 3-0, we also extend their work to non-Lambertian scenes for the second stage of AIBR.

Active capturing for image-based rendering is very much related to the so-called "next best view" (NBV) problem in the automated surface acquisition literature [16][17]. Both AIBR and NBV try to determine the next best position (and orientation) of the sensor so that minimum number of images is required, or best performance is achieved at given number of images. However, they are yet very different problems. A range sensor is often used in NBV problems, while we take pictures for the scene in AIBR. The benefit of taking one more range image can be well predicted based on the holes in the current surface model and the sensor position/orientation, but the advantage of taking one more image for IBR is not obvious. Most importantly, the final goal of NBV problem is to recover the object surface. In AIBR, we do not care too much about how accurate the voxel model is for the scene. Instead, we care about the rendering image quality.

AIBR is also related to the active learning approach in the machine learning literature [18][19][20]. For many types of machine learning algorithms, one can find the statistically "optimal" way to select the training data. The pursuing of the "optimal" way by the machine itself was referred to as *active learning*. For example, a classification machine may determine the next training data as the one that is most difficult for it to classify. AIBR has the same favor because it always puts the next camera to places where it has the worst rendering quality.

3. Active Capturing for Image-Based Rendering

As we mentioned in Section 1, we allow the camera to be anywhere on the camera plane during the capturing. As shown in Figure 2, assume that the capturing cameras are within a rectangular range determined by (0,0)

and (s_{max}, t_{max}). We initialize the active capturing process by a reasonably dense uniform sampling. Define *quadruple* as the group of four images that form a unit rectangle. Each time when we capture some new images, we split one of the quadruples into four. In the example shown in Figure 2, five new images are taken during the split. The organization of the images in AIBR is thus very similar to a quadtree. This data structure is very helpful for the capturing and rendering process, which will be addressed later. AIBR recursively performs the above split until the constrained number of images is reached or some rendering quality criteria are satisfied.

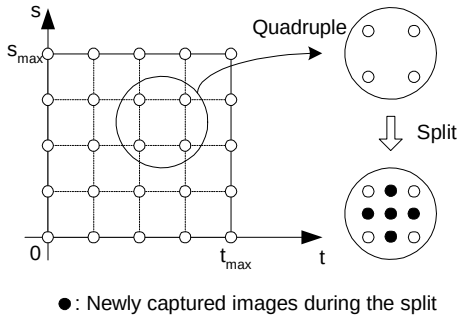


Figure 2 The capturing pattern of AIBR.

In this section we first discuss the color consistency criterion, which helps us to estimate the rendering image quality for each quadruple given a voxel model of the scene. When the scene geometry is known, AIBR is as simple as iteratively choosing the worst quality quadruples and splitting them. When the geometry is unknown, we propose a local-consistency voxel coloring approach for generic scenes including non-Lambertian ones. The full two-stage AIBR algorithm is the combination of the above components.

A. The color consistency criterion

When the scene is Lambertian, light rays from the same object surface point should have the same color. This is referred as the *color consistency criterion* in this paper. Color consistency has found many applications in geometry reconstruction problems, such the voxel coloring algorithm [15] and various stereo algorithms [21][22].

We notice that color consistency is also critical for image-based rendering. Consider the depth-driven rendering approach similar to [12], as shown in Figure 3. C_1, C_2, …, C_6 are views that have been captured. To render the virtual view C, we split it into many light rays. For each light ray, e.g., CP, we trace it to a point on the object surface, in this example, P. We then project point P to all the nearby captured views. The intensity of the light ray CP is approximated by weighted interpolation of the nearby light rays such as C_3P, C_4P, etc. If the scene is Lambertian, the nearby light rays should have the same color. That is, they should be color consistent. In practice, these light rays may have different colors for many reasons, such as the poor depth information of P, the non-Lambertian property of the surface, occlusions, sensor noise, etc. Most ghosting or aliasing effects

in IBR rendering are due to the violation of the color consistency principle during the interpolation. Fortunately, these bad effects can always be eliminated by taking more sample images around *C*.

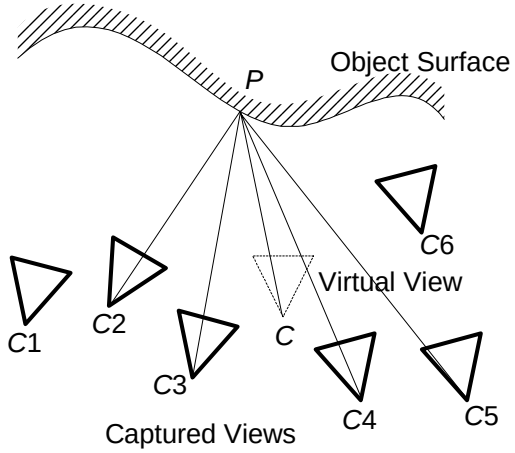


Figure 3 The depth-driven IBR rendering scheme.

The above discussion implies that color consistency verification can be a useful tool to estimate the rendering image quality. Consider an arbitrary quadruple q in Figure 2. Let I_j, j=1,2,3,4 be the four corner images of q. Let V be the voxel model of the scene, and v_i, i=1,2,…,N be its occupied voxels sorted layer by layer from near to far. We measure the inconsistency score of q as follows. Take the occupied voxels v_i, i=1,2,…,N in sequence and project them to the four corner images. The variances of the projected pixel colors are accumulated. The final inconsistency score is defined as the average variance of all the visible voxels. To handle possible occlusions, we maintain mask images M_j, j=1,2,3,4 as in the standard voxel coloring algorithm. Whenever a voxel is projected to the images, the projected pixels are marked in the mask images. If later another voxel is projected to the same position in a certain image, it has to be ignored because the latter voxel is occluded by the previous one from that viewpoint. The pseudo code of the above algorithm is shown in Figure 4.

In the above algorithm, we introduced *single projection penalty* for voxels that have only one valid projection. This is because we are not very confident about the color of that voxel if we have only one observation. The penalty is heuristically determined as a constant value in the current implementation. The returned inconsistency score is the average variance of the projected pixel colors for all the visible voxels. Clearly, the higher the inconsistency score, the worse the rendering quality. Measures other than variance can also be easily incorporated into the algorithm, e.g., χ^2 statistics [23], F distribution [23][24], etc.

```
function score = InconsistencyScore (q)
  sum = 0, count = 0;
  Reset the mask images M_j to 0;
  for i=1,…,N
    project v_i to I_j, j=1,2,3,4;
    check the validity of the projections by M_j;
    switch (# of valid projections)
      case 0: continue;
      case 1:
        sum += single projection penalty, count++;
      case 2,3,4:
        sum += variance of the valid projected pixels;
        count ++;
    end switch
    Mark the valid projected pixels in M_j;
  end for
  score = sum/count;
  return score;
```

Figure 4 Algorithm for estimating the rendering quality of light rays passing through quadruple *q*.

Notice that if we strictly follow the rendering scheme in Figure 3, the four corner images may not be the closest captured images when the rendered light ray is inside the rectangle of the quadruple. Neighboring quadruples may have been split and there may be captured images at the edges of the rectangle. Although we do consider such cases during the rendering, we ignore them in the performance estimation stage for simplicity. Another concern is whether the algorithm will over-estimate the rendering quality. The answer is positive, because if the intensities of the light rays changes fast within the quadruple, our estimation can be very wrong. This is similar to the aliasing effect in signal processing. To reduce the risk caused by over-estimation, we initialize our active capturing with a uniform sampling that is reasonably dense. An alternative solution might be to integrate the rectangle area of the quadruple into the quality measurement. When the area is relatively large, there is higher chance for over-estimation.

B. Active capturing when geometry is known

Although getting the geometry information about the scene is very difficult, in this subsection we assume the geometry is known. For example, it can be obtained from the images captured so far. Subsection 0 will present

a voxel coloring algorithm for generic scenes. In worst case, we may simply assume that the scene lies at a constant depth. This is widely used in the literature when the geometry is hard to recover [4][5][12].

```
function ActiveIBR_KG ()
start:
  for all the quadruples q_k, k=1,2,…,K
    score_k = InconsistencyScore(q_k)
  find the quadruple that has max(score_k) and split it
  if (max #of images reached or max(score_k)<T)
    return;
  else
    go to start;
```

Figure 5 AIBR for known geometry.

When the geometry is known, active capturing is very straightforward. We measure the inconsistency score for each quadruple with the algorithm in Figure 4, and then split the quadruple that has the highest score or worst estimated rendering quality. The algorithm *ActiveIBR_KG* (where KG stands for Known Geometry) is shown in Figure 5, where q_k, k=1,2,…,K represent all the current existing quadruples.

In Figure 5, the algorithm runs recursively. There are two stopping criteria. One is when the number of captured images reached a predefined limit. The other is when the inconsistency scores of all the quadruples are all less than a certain threshold, which guarantees the rendering quality to some degree. As the captured images are organized in a quadtree manner, in each loop only the newly generated quadruples need to be measured for their rendering quality, which is very time-efficient.

C. Voxel coloring for non-Lambertian scenes

Voxel coloring [15] has been a very useful tool for recovering scene geometry from a set of images. A survey on various volumetric scene reconstruction algorithms including voxel coloring can be found in [25]. A common assumption in these algorithms is that the scene is Lambertian, or near-Lambertian. Various color consistency measures [23][24] have been proposed under this assumption. When the scene has a lot of noise or is non-Lambertian, variable threshold may be used, or we can apply some probabilistic consistency function for voxel coloring [26][27]. The problem with the probabilistic approaches is that they assume light rays from the same surface point follows a Gaussian distribution. Although it might be able to handle noises well, the Gaussian distribution is not a reasonable assumption for highly reflective surfaces. Moreover, these methods are very time-consuming compared with simple color consistency measures in the Lambertian case.

There is a difference between the voxel coloring algorithms in the literature and the one we want to propose for active IBR. In all the previous work, the goal is to find the 3D model of the scene as good as possible. In IBR, we recover the 3D model for depth-driven rendering. Our goal is to have the best rendering quality, but not to find the best 3D model. *With rendering in mind*, we define a color consistency measure based on local verification. For each voxel being tested, we claim it to be occupied when for all the quadruples the voxel is color consistent. As light rays from the same scene surface point to the images in one quadruple are often along very similar directions, we can assume that they have similar colors. Thus all the old simple color consistency measures can be used here. Since during the rendering we interpolate light rays only from neighboring light rays within a quadruple, the 3D model obtained from our voxel coloring algorithm can guarantee a good rendering quality. We show our algorithm in Figure 6. The last stage of our voxel coloring algorithm is to add a plane at the maximum depth of occupied voxels. This is to avoid holes during the IBR rendering.

```
function Voxelcoloring ()
  for all the possible voxels from near to far
    project it to q_k, k=1,2, …, K
    measure color consistency for each q_k
    if for all q_k it is color consistent {
      mark the voxel as occupied;
      do supplemental things such as handling mask images;
    }
  end for
  add a plane at the maximum depth of occupied  voxels;
```

Figure 6 Voxel coloring for non-Lambertian scenes.

Similar to the original voxel coloring algorithm, our algorithm has a systematic bias to small depth voxels. Consider an extreme case where we have taken many sample images about the scene, the resultant voxel model may simply be a plane at the minimum depth. However, the bad voxel model will not hurt our rendering quality, as IBR can run well as long as the number of images and the geometry information jointly satisfy the sampling curve requirement in [13]. In fact, the whole process of AIBR will help to find the best balance between the two factors.

D. A recursive approach for general scenes

Our full active IBR algorithm runs active IBR for known geometry and voxel coloring recursively. We initialize the capturing process by a reasonable dense uniform sampling.

We then apply voxel coloring and get a 3D voxel model. With the voxel model, we will be able to find which quadruple to split with an algorithm similar to that in *ActiveIBR_KG*. After the splitting, we may continue splitting or applying voxel coloring again. The whole process loops until the limit of the number of images is reached or all the quadruple has a good color consistency. The algorithm is shown in Figure 7.

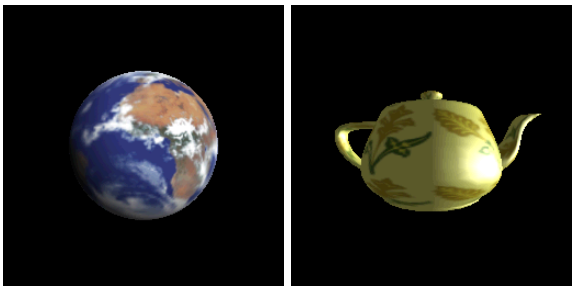
```
function ActiveIBR ()
  uniformly taking images as initialization.
loop:
  Voxelcoloring();
  for all the quadruples q_k, k=1,2,…,K
    score_k = InconsistencyScore (q_k)
  find the quadruple that has max(score_k) and split it
  if (max #of images reached or max(score_k)<T)
    return;
  else
    go to loop;
```

Figure 7 The full active IBR algorithm.

The voxel coloring stage may take a long time to execute. Therefore we may not want to do voxel coloring for every split. At a certain stage, voxel coloring may actually make the geometry worse due to the systematic bias to small depth voxels. Therefore after a certain period we should leave voxel coloring algorithm out of the iteration.

4. Experimental Results

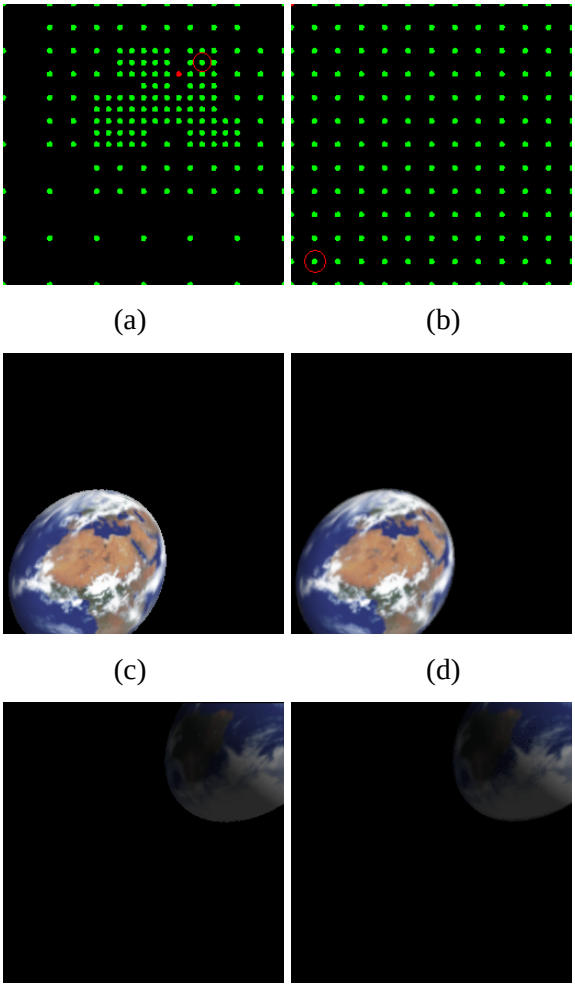
We test our algorithm on two synthetic scenes, *Earth* and *Teapot*. They are shown in Figure 8 (a) and (b), respectively. *Earth* is a near-Lambertian scene, and we assume that its geometry is known. *Teapot*, on the other hand, is specular and illuminated by a spot light. The voxel model of *Teapot* is unknown, and we will use our voxel coloring algorithm to recover it during active capturing process.



(a) (b)

Figure 8 The two test scenes. (a) Earth. (b) Teapot.

We first apply the AIBR algorithm for known geometry (Figure 5) to *Earth*. The geometry is described by a known 96×96×64 voxel model. We initialize our algorithm by a 7×7 uniform sampling, and the overall number of images is limited to be less than or equal to 169. The result is compared to a 13×13 uniform IBR (UIBR) sampling approach.



(a) (b)

(c) (d)

(e) (f)

Figure 9 The final camera map of AIBR and UIBR and some sample images.

Figure 9 (a) shows the final camera map of AIBR. For comparison, (b) shows that of UIBR. Each dot represents a camera being there and taking one image. It can be observed that active IBR puts more cameras at the top-right portion of the camera plane. Figure 9 (c) is an example view captured in AIBR (red circled in (a)). UIBR did not sample that view. Figure 9 (d) is what can be rendered from the sampled images in UIBR. As a comparison, Figure 9 (e) is a view captured in UIBR (red circled in (b)). It is not captured in AIBR but can be rendered as (f). Obviously we would prefer to sample (c) instead of (e) because the quality degradation from (c) to (d) is more obvious than that from (e) to (f). Therefore AIBR made the right decision.

To measure the improvement of active IBR over the traditional uniformly sampled IBR, we employ two objective measures. The first is the worst-case quality. From the quadruples formed by both approaches, we render the virtual views at their centers. As the center views are the farthest from the sampled images, most likely they will have the worst quality. Our first measure is the average peak signal-to-noise ratio (PSNR) of the worst 30 center views. Notice that we are able to measure the PSNRs because we are rendering 3D models and we have the real rendered images as our ground-truth. The second measure is the PSNR variance of rendered images. We randomly render 1000 images on the camera plane and measure the variance of the PSNRs. The results are shown in Table I. It can be observed that active IBR has a better worst-case quality and a smaller variance, which is what we want.

Table I: Comparison between UIBR and AIBR on scene *Earth*.

	UIBR	AIBR
Avg. PSNR of 30 worst center views	32.8dB	33.2dB
PSNR Var. of 1000 rendered images	9.87	6.82

We next show some experimental results on voxel coloring. *Teapot* has a highly reflective surface, which is a disaster to the traditional voxel coloring algorithms. We represent the geometry of *Teapot* with a 192×192×128 voxel model. 7×7 images and 13×13 images are uniformly sampled on the camera plane. We then apply the traditional voxel coloring and our algorithm on the images. The resultant voxel models are shown in Figure 10 (a), (b), (c) and (d). Notice that in both sampling density, our algorithm gives better results. More importantly, the traditional voxel coloring does not improve when the number of sample image increases, while our result for 13×13 images is much better than 7×7 samples.

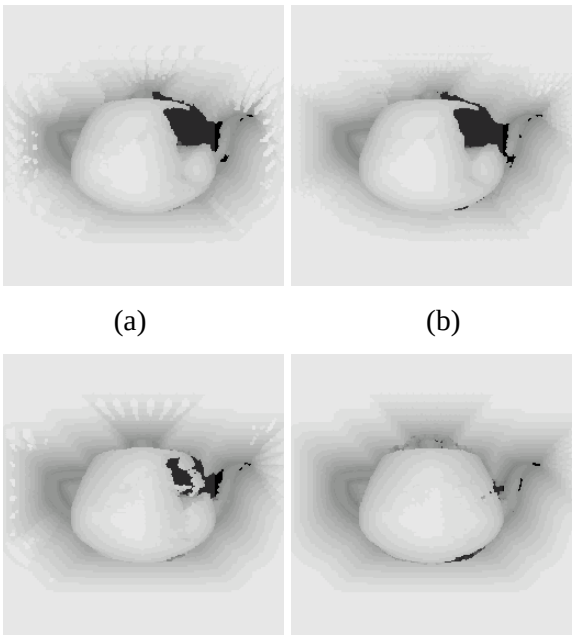


Figure 10 Voxel coloring results for Teapot. (a) Traditional method, 7×7 sample images. (b) Traditional method, 13×13 sample images. (c) Proposed method, 7×7 sample images. (d) Proposed method, 13×13 sample images.

We finally show some rendered images for the *Teapot* scene. The comparison is between uniform sampling with optimal constant depth and active IBR with the new voxel coloring algorithm. The recursive process in Figure 7 is applied for active IBR. The overall number of images is limited as 169, and the active IBR algorithm is initialized with 7×7 uniform samples. Active IBR does geometry refinement for every 20 newly captured images. Three example rendered images are shown in Figure 11 (a1-a3), (b1-b3), respectively. Notice the flower texture on the teapot. It is very obvious that the images rendered by our algorithm are much better than that by the traditional method.

5. Conclusions and Future Work

In this paper, we introduced active capturing for image-based rendering, which poses capturing cameras intelligently based on the estimation of rendering quality. We also proposed a new voxel coloring algorithm that can handle non-Lambertian scenes. The combination of these two produces better results than the tradition IBR approach.

The active capturing algorithm for IBR is a framework rather than a specific algorithm. It basically employs rendering quality estimation, capturing and geometry refinement recursively. Quadruple is a special case of neighborhoods. If we do not have a quadtree structure during the capturing, all the discussions are still valid for neighborhoods. Similarly, although voxel-based representation is used in our algorithm, other geometry representation can also apply.

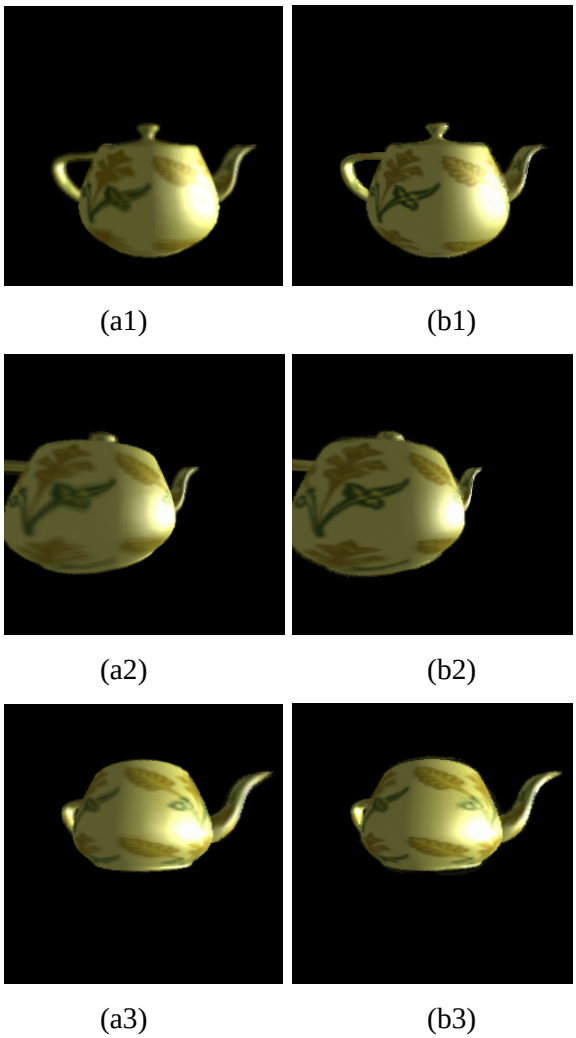


Figure 11 Comparison of rendered images. (a1-a3) UIBR rendered at constant depth. (b1-b3) AIBR rendered with the reconstructed voxel model.

There are yet some limitations on our algorithm. The quadtree structure makes our algorithm elegant but it also hurts the performance to some extent because the capturing pattern is constrained. We also need to know exactly where the camera is, which is difficult in real applications. We are currently working on applying the AIBR algorithm in real situations.

References

[1] S. B. Kang. "A survey of image-based rendering techniques", *VideoMetrics, SPIE* Vol. 3641, pages 2-16, 1999.

[2] L. McMillan and G. Bishop, "Plenoptic modeling: an image-based rendering system", *Computer Graphics (SIGGRAPH'95)*, pp. 39-46, Aug. 1995.

[3] M. Levoy and P. Hanrahan, "Light field rendering", *Computer Graphics (SIGGRAPH'96)*, pp. 31, Aug. 1996.

[4] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F. Cohen, “The Lumigraph”, *Computer Graphics (SIGGRAPH’96)*, pp. 43-54, Aug. 1996.

[5] H.Y. Shum and L.-W. He, “Rendering with concentric mosaics”, *Computer Graphics (SIGGRAPH’99)*, pp.299-306, Aug. 1999.

[6] S. Chen and L. Williams, “View interpolation for image synthesis”, *Computer Graphics (SIGGRAPH’93)*, pp. 279-288, Aug. 1993.

[7] S. M. Seitz and C.M. Dyer, “View morphing”, *Computer Graphics (SIGGRAPH’96)*, pp. 21-30, Aug. 1996.

[8] M. Pollefeys, "Self-calibration and metric 3D reconstruction from uncalibrated image sequences", *Ph.D. Thesis*, ESAT-PSI, K. U. Leuven, 1999.

[9] L. McMillan, “An image-based approach to three-dimensional computer graphics”, *Technical Report, UNC Computer Science TR97-013*, 1999.

[10] J. Shade, S. Gortler, L. W. He, and R. Szeliski, “Layered depth images”, *Computer Graphics (SIGGRAPH’98)*, pp. 231-242, July 1998.

[11] P. E. Debevec, C. J. Taylor, and J. Malik, “Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach”, *Computer Graphics (SIGGRAPH’96)*, pp. 11-20, Aug. 1996.

[12] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, “Unstructured Lumigraph rendering”, *Computer Graphics (SIG-GRAPH’01)*, pp 425-432, Aug. 2001.

[13] J.X. Chai, X. Tong, S.C. Chan and H. Y. Shum, “Plenoptic sampling”, *Computer Graphics (SIGGRAPH’00)*, pp.307-318, July 2000.

[14] C. Zhang and T. Chen, “Generalized Plenoptic Sampling”, *Carnegie Mellon University Technical Report, AMP01-06*.

[15] S. M. Seitz and C. R. Dyer, “Photorealistic Scene Reconstruction by Voxel Coloring”, *CVPR 1997*, pp. 1067-1073.

[16] R. Pito, “A Solution to the Next Best View Problem for Automated Surface Acquisition”, *IEEE Transactions on PAMI*, vol. 21, Oct. 1999.

[17] M. K. Reed, “Solid Model Acquisition from Range Images” *Ph.D. thesis*, Columbia University, 1998.

[18] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, “Active Learning with Statistical Models”, *Journal of Artificial Intelligence Research*, pp. 129-145, Vol. 4, 1996.

[19] A. Krogh and J. Vedelsby, “Neural network ensembles, cross validation, and active learning”, In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, Cambridge, MA, 1995. MIT Press.

[20] M. Hasenjäger, H. Ritter, and K. Obermayer, “Active learning in self-organizing maps”, In E. Oja and S. Kaski, editors, *Kohonen Maps*, pp. 57-70. Elsevier, Amsterdam, 1999.

[21] I. Cox, S. Hingoraini, S. Rao, “A maximum likelihood stereo algorithm”, *Computer Vision and Image Understanding*, Vol. 63, No. 3, pp. 542-567, May, 1996.

[22] S. Roy and I. J. Cox, “A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem”, *Int. Conf. on Computer Vision (ICCV'98)*, pp. 492-499, Jan. 1998,

[23] S.M. Seitz and C.M. Dyer. “Photorealistic scene reconstruction by voxel coloring”, *Int. Journal of Computer Vision*, Vol. 35, No. 2, pp.1067–1073, 1999.

[24] A. Broadhurst and R. Cipolla, “A statistical consistency check for the space carving algorithm”, *Proc. 11th British Machine Vision Conference*, volume I, pp.282–291, Bristol, Sep. 2000.

[25] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer, “A survey of methods for volumetric scene reconstruction from photographs”, *Proceedings of the Joint IEEE TCVG and Eurographics Workshop (VolumeGraphics-01)*, pp.81--100, June, 2001.

[26] A. Broadhurst, “A Probabilistic Framework for Space Carving”, *Ph.D. Thesis*, University of Cambridge, 2001.

[27] A. Yezzi, G. Slabaugh, A. Broadhurst, R. Cipolla, R. Schafer, “A Surface Evolution Approach to Probabilistic Space Carving”, *The 1st International Symposium on 3DProcessing, Visualization, and Transmission (3DPVT) 2002*.