Image-Based Objects

Manuel M. Oliveira and Gary Bishop[*]

University of North Carolina at Chapel Hill

Abstract

We present a compact, image-based representation for three-dimensional objects with complex shapes that can be rendered with correct perspective from arbitrary viewpoints using a list-priority algorithm. Objects are represented by six layered depth images sharing a single center of projection. They can be scaled, and freely translated and rotated, being used as primitives to construct more complex scenes. We also present a new list-priority algorithm for rendering such scenes and a back face culling strategy for a class of image-based objects.

We demonstrate these concepts by constructing image-based representations from both synthetic and real objects, and rendering them at interactive rates on a PC. Due to their minimum storage requirements and rendering simplicity, image-based objects can find potential uses in games, virtual museum applications, and web catalogs.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation – Display Algorithms; I.3.7 [Three-dimensional Graphics and Realism]: Visible Surface Algorithms.

Additional Keywords: image-based rendering, image-based objects, 3-D warp.

1 INTRODUCTION

Image-based representations of objects are currently used in computer games and virtual museum applications, and there is some potential demand for web-based shop catalogs. An ideal object representation for such applications should preserve the original appearance of the objects, be able to be manipulated interactively, and visualized from arbitrary viewpoints. Also it should be compact enough to be sent through a network, and rendered at reasonable frame rates using non-specialized graphics hardware.

This paper presents a new compact image-based representation for three-dimensional objects with complex shapes that can be rendered with correct perspective from arbitrary viewpoints using a list-priority algorithm based on McMillan and Bishop's [7] occlusion compatible order. In our approach, each object, called an Image-Based Object (IBO), is represented by six layered depth images (LDIs) [13] that share a single center of projection (COP). IBOs can be scaled, arbitrarily translated and rotated, and can be regarded as primitives to construct more complex scenes. We also present a new list-priority algorithm for rendering dynamic scenes composed of IBOs, given some spatial constraints among the objects. We demonstrate these concepts by constructing image-based representations from both synthetic and real objects and by implementing a system prototype that can render IBOs at interactive rates on a PC.

[*] CB #3175 Sitterson Hall, Chapel Hill, NC, 27599-3175
oliveira@cs.unc.edu, gb@cs.unc.edu

2 RELATED WORK

Dally et al [2] use a sampling sphere around a target object to take multiple views of it. Such images are stored in a data structure called a delta-tree that divides the (θ, ϕ) space into square regions. During rendering time, the four corners of the region enclosing the desired viewpoint are used for reconstruction [2]. Since the COPs of all reference images are constrained to be on the sphere, it is not possible to completely sample the surface of objects with arbitrary shapes. The multiple views have different centers of projection and a z-buffer is required to eliminate hidden surfaces.

Levoy and Hanrahan [6] and Gortler et al [3] represent objects as collections of images obtained from rectangular grids placed around the objects. At rendering time, the images in the database are resampled to produce an interpolated view of the object. This approach requires a large number of images, and putting multiple such representations in the same scene is not straightforward.

Pulli et al [10] use color images and dense range maps to reconstruct sparse triangle meshes associated with real objects. The color images are used as texture maps and applied to the meshes. In order to reconstruct a new view of an object, the meshes corresponding to the three closest original viewpoints are blended on a per pixel basis and z-buffered in software.

Schaufler [12] uses several layers of texture-mapped quadrilaterals to render 3-D objects. Objects have associated textures augmented with depth on a per pixel basis. The depth value corresponding to a pixel is used to select the quadrilateral the pixel is mapped to. A scene is rendered warping the textures of all objects individually, which are z-buffered to produce the final image.

Grossman and Dally [4] represent objects as dense sets of surface point samples which contain color, depth, and normal information. Such point samples are rendered using a hierarchy of z-buffers to detect tears, and Phong shading and shadow generation are supported.

Multiple-center-of-projection images [11] can represent objects as sequences of one-dimensional images acquired along a continuous path. Such a representation provides connectivity information among adjacent samples, and allows different parts of the object/scene to be sampled at different resolutions. Since samples are acquired from different COPs, visibility is determined using a z-buffer.

3 RENDERING OBJECTS USING A LIST-PRIORITY ALGORITHM

McMillan and Bishop [7] presented a list-priority solution for the visibility problem in the context of their image warping framework. Figure 1 illustrates the basics of their method using planar images. Vectors a_i, b_i and c_i define a projective pinhole camera. a_i and b_i are orthogonal and form a basis for the plane of image *i*. The lengths of these vectors are the width and height of a pixel in the Euclidean space, respectively. C_i is the COP of the associated camera. c_i is a vector from the COP to the origin of the image plane. The projection of one camera's COP into the image plane of the other is called an *epipole* (Figure 1). An occlusion compatible order for a planar reference image can be summarized as [8]: if the COP of the desired view is behind the reference image plane, the samples of the reference image are warped from its epipole towards the borders of the image; otherwise, they are warped from the borders towards the epipole.

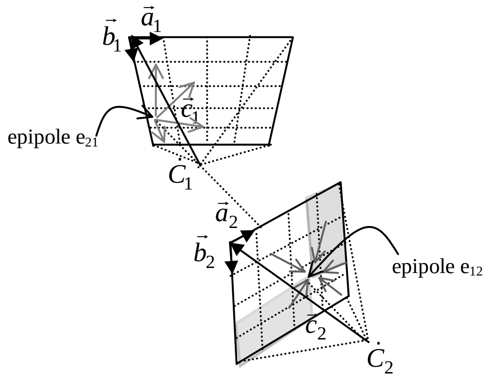
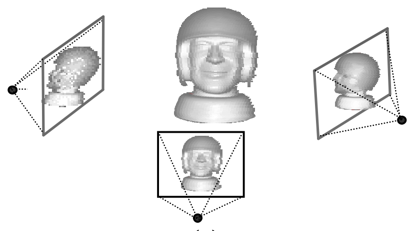


Figure 1. Two projective pinhole cameras. The epipoles divide the images into at most four regions (shaded in image 2). The gray arrows specify the order in which the samples should be warped, if the image is the reference one.

McMillan and Bishop's algorithm cannot be used to warp multiple images acquired from different COPs simultaneously. An important observation, however, is that their algorithm can still be used if all images share a common COP. In this case, one only needs to specify the order in which the images must be warped, which changes with the desired viewpoint. Unfortunately, it is not possible to sample the whole exterior surface of a three-dimensional object from a single COP.

3.1 Building Image-Based Objects

One solution to this problem is to put the shared COP inside the object. Although this is not directly realizable for real objects, this idea gives us a good framework to think about the problem. A similar result can be obtained by acquiring multiple views of the object, and resampling them from a single COP, as illustrated in Figure 2. The depth values associated with the pixels are used to project all samples back to 3-D. Once these samples have been registered, they are reprojected into perpendicular image planes sharing the same COP (Figure 2b). Notice that, although a cubic arrangement is shown in Figure 2, any parallelepiped would work as well. Also, the object does not need to be completely inside the parallelepiped. In fact, as the box gets smaller, the more uniform the object re-sampling becomes, specially for elongated shapes. For instance, the rectilinear box used to create the IBO shown in Figure 13 was very small compared to the actual object size, and was located almost completely inside the statue's belly.


(a)

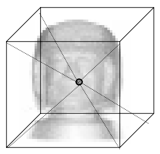

(b)

Figure 2. (a) Views of an object acquired from multiple COPs. (b) The samples are registered and will be reprojected onto perpendicular image planes (represented by the faces of the cube) sharing a single COP.

During the resampling process, multiple non-redundant samples falling along the same ray are preserved. Thus, each image-based object is represented by six LDIs, which are stored as linear arrays for efficient warping.

Notice that although our representation makes use of LDIs, the two concepts are quite different. LDIs were introduced to minimize disocclusion problems that occur when warping depth images [13]. This is achieved by allowing a view of the scene to contain multiple samples along each ray. Whereas LDIs can be warped in occlusion compatible ordering, they are only effective if the desired view is in a certain neighborhood of the LDI's COP. An apparent similarity between both concepts is the use of a cube when generating LDIs from ray traced scenes. In this case, the purpose of such a cube is to define the region of interest in which the viewer will be allowed to move when exploring the scene [13]. In our approach, the goal is to produce arbitrary views of a three-dimensional object from a fixed set of six images warped in occlusion compatible order. The parallelepiped (shown in Figure 2) is used for two reasons: first, to provide a surface topologically equivalent to a sphere, for which an occlusion compatible ordering is known to exist [8]. Secondly, it can be decomposed into parameterized planar regions (faces of the parallelepiped), for which a warper can be implemented efficiently.

Given the resolutions (possibly different) of the planar images associated with the faces of the parallelepiped, each original sample is mapped to the closest pixel of the image covering the corresponding region of space. The higher the resolution the smaller the reprojection error. Alternatively, the surfaces of the objects can be reconstructed and resampled using a regular grid at the corresponding image planes. In all examples shown in this paper, samples were mapped to the center of the closest pixel they project to.

A sample is considered to be redundant if it is closer than a pre-defined threshold in 3-D (Euclidean distance) to another sample from a different original image, and both have similar colors. Redundant samples are eliminated during the construction of the LDIs. The preprocessing time associated with the construction of the six LDIs corresponding to the IBO shown in Figure 15 was about 5 seconds on a HP workstation, after which the LDIs are saved in disk and are ready for future use. In our current implementation, the choice of which samples are preserved (among the redundant ones) is arbitrary. A better solution seems to base such a decision on the angle between the ray from the IBO COP to the sample, and the sample normal. Such a normal can be approximated using the neighborhood of the sample in the corresponding original image [9], as part of this preprocessing.

3.2 List-Priority Rendering

Consider a spherical reference image and a desired viewpoint. The viewing ray passing through the center of the spherical image intersects its surface at two points. The one closest to the viewer (or behind him/her, if the viewer is inside the sphere) is called the *positive epipole* (e+); The other one is called the *negative epipole* (e-). An occlusion compatible order for spherical reference images consists of warping samples from the negative epipole towards the positive one [8].

Since the IBO representation is topologically equivalent to a sphere, each IBO can be warped using an adaptation of McMillan and Bishop's algorithm for spherical reference images. The IBO COP is defined as the intersection of any two of the four diagonals of the parallelepiped. Given such a configuration, the following properties can be observed:

- Although the six planar images share a common COP, they are still independent from one another. Therefore, the definition of regions is independently established for each image, using the regular region split procedure (Figure 1);
- Since the IBO COP is at the intersection of the diagonals, the positive and the negative epipoles fall within opposite faces;
- There is no redundancy among images, *i.e.*, no sample is seen in more than one face;
- The whole field of view is covered;
- Once a relative order among the planar images has been established, each image can be independently warped using the conventional 3-D image warping algorithm [7];

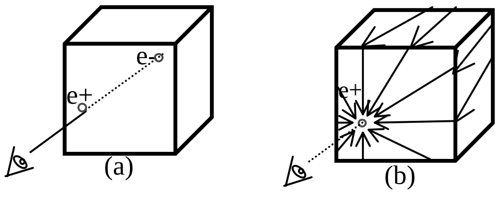


Figure 3. Epipolar Geometry of an IBO. (a) The segment containing the desired COP and the IBO COP (center of the box) intersects opposite faces at e+ (positive epipole), and e- (negative epipole). (b) Projected flow lines (arrows) defining an occlusion compatible ordering for the whole object.

Figure 3 illustrates the epipolar geometry for a cubic IBO. The line connecting the desired and the IBO COPs intersects the cube at opposite faces[1], and defines an occlusion compatible order for warping the whole object. The face containing the negative epipole (e-) (Figure 3) must be warped first, while the face containing the positive epipole (e+) must be warped last. The arrows in Figure 3b are the projected flow lines representing the occlusion compatible order. Consider the same cube split into six pyramids with apices at the IBO COP, as shown in the Figure 4. Let's call the faces containing the positive and negative epipoles, F, and K, respectively. Notice that this classification is relative to desired view position. The other two pairs of opposite faces are called (A, A'), and (B, B') (Figure 4). The following theorem defines orders in which the six faces can be independently warped in occlusion compatible order. A proof of the theorem is presented in Appendix A.

[1] The cases in which the line intersects edges or vertices are treated similarly.

Theorem: Let B' be the base of pyramid $P_{B'}$ that is intersected by the segment connecting the positive epipole and its parallel projection into K. Then, warping the faces of the cube in the order (K, B, A, A', B', F), or (K, B, A', A, B', F) produce correct visibility from the desired view position.

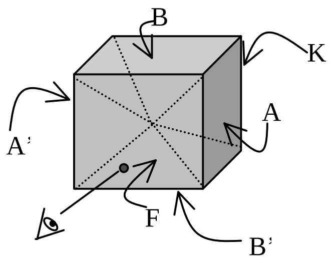


Figure 4. Faces of the parallelepiped are labeled with respect to the desired viewpoint. F contains the positive epipole (e+), whereas K contains the negative epipole (e-).

Since the set of rays emanating from the object COP covers a solid angle of 4π steradians, some care should be taken in order to guarantee that multiple samples along a ray are always warped from back to front with respect to the desired view. One way to define such an order is to compute the smallest angle between the two vectors from both COPs to the furthest sample along the ray in question. If the angle is less than 90 degrees, the samples are warped from farthest to closest (with respect to the IBO COP); otherwise, they are warped from closest to farthest. Notice, however, that such a procedure requires the knowledge of the ray direction associated with the projection of the sample in the desired image plane, which is only known after the actual warping. In order to avoid an extra warping step just to compute such a direction, we approximate the desired ray using the desired image plane normal. This way, the order in which samples are warped is established by a dot product. Although this is only an approximation, it works very well in practice.

In the case of objects whose representations are topologically equivalent to spheres (genus zero) and present good aspect ratio (*i.e.*, they are not elongated with respect to any particular dimension), the warping of the image containing the negative epipole can be omitted (Figure 17). This optimization is analogous to back face culling used in polygonal computer graphics. In practice, we observed speedups varying from 19% to 22% due to its utilization.

Objects whose surfaces have genus greater than zero (*e.g.* torus) are represented by assigning zero disparity to the LDIs' rays that do not hit the object's surface. During warping time, such rays are ignored allowing the background to become visible.

3.3 Transformations

Geometric transformations such as translation, rotation, and scaling can be easily applied to IBOs. Since all six LDIs are defined with respect to a single COP, translations are obtained simply by translating the object COP. Rotations are obtained by rotating the a, b, and c vectors that define the pinhole camera

(Figure 1) of all six LDIs around the desired axis. An IBO is scaled by a factor *s* by multiplying these vectors by *s*.

4 SCENES FROM IBOS

Often, multiple objects are used together to create more complex scenes. This section explains how IBOs can be used to achieve similar results.

Given some constraints on the spatial relationship among the objects, the whole scene can be rendered using an extension of the same algorithm used to render objects. Thus, if there are no inter-penetrations between any pair of IBOs bounding spheres, then for any desired view there is at least one serial order in which the objects can be warped that produces correct visibility and does not require depth comparison.

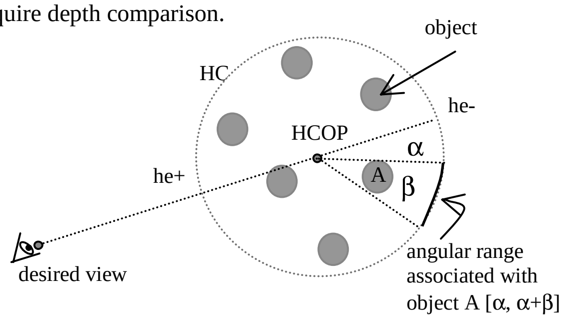


Figure 5. Epipolar geometry of a scene with respect to the desired view. Small circles represent IBOs bounding circles (in 2-D). HC (hypothetical circle), HCOP (hypothetical COP). All angles are positive.

The algorithm presented here in 2-D, for simplicity, provides a way to obtain one such order. Given a set of objects, compute a hypothetical COP for the scene (HCOP) as the average of all objects COPs. If the derived HCOP does fall inside any object's bounding circle, move it to avoid this situation. Define HC, a hypothetical circle whose center is at HCOP. Given an arbitrary desired view, he- and he+ are the hypothetical negative and positive epipoles on HC induced by the desired view (Figure 5). The segment he+/he- divides the circle into two semi-circles. Next, compute the angular range (with respect to the segment HCOP/he-) associated with each object of each semi-circle. Notice that there is no need to actually compute he+ or he-, since the vector from the viewpoint to HCOP can be used to compute angular ranges. he+ and he- are used here for didactical reasons.

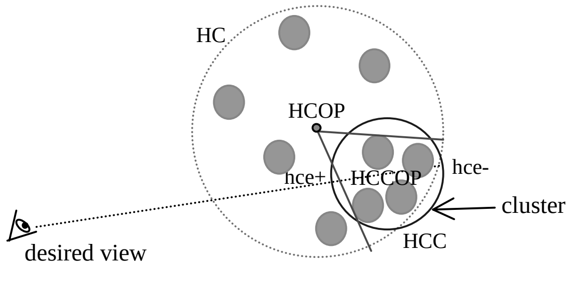


Figure 6. Recursive application of the algorithm to solve visibility inside a cluster of overlapping angular ranges.

A group of objects whose angular ranges overlap form a cluster (Figure 6). The criterion for defining clusters is transitive, *i.e.*, if the range of object A overlaps the range of object B, and the range of object B overlaps the range of object C, then A, B, and C belong to the same cluster.

The objects are sorted by semi-circle (in which they fall in) and by the lower bound of their angular ranges (notice that all angles are positive – Figure 5). The elements of the sorted list are visited and every time we reach an object that does not belong to a cluster (*i.e.*, whose angular range does not overlap with the next object's angular range), it is added to the end of a list (originally empty). If the object belongs to a cluster, the algorithm is recursively applied to the cluster itself. In such a case, an HCCOP (hypothetical cluster COP), an HCC (hypothetical cluster circle), and a pair of epipoles (hce- and hce+) are defined for the cluster (Figure 6). At the end, the constructed list defines an order that produces correct visibility from the desired viewpoint. The faces of individual objects are warped using the order defined by the theorem presented in section 3.2. Figure 7 presents a pseudocode for the ordering procedure described. In the first call hcop = HCOP and rendering_list is empty.

```
Order (viewpoint, hcop, object_list, rendering_list)
{
    for each object in object_list do
        compute object's semi-circle and angular range;
    sorted_list ← sort_by_semi-circle_then_by_
                      lower_bound_angular_range (object_list);
    while sorted_list not empty do {
        object ← first (sorted_list);
        if ( not overlap (object, next(sorted_list)))
            append (rendering_list, object);
            remove (sorted_list, object);
        else
//          find all objects whose ranges overlap with
//          object's range
            cluster ← overlap_range (sorted_list, object);
            cluster_hcop ← average_centers (cluster);
            Order (viewpoint, cluster_hcop, cluster,
                      rendering_list);
            for each object in cluster do
                remove (sorted_list, object);
        endif;
    }
}
```

Figure 7. Pseudocode for the ordering procedure.

A cluster covers a whole angular range in HC. The order in which its elements should be rendered is completely specified by them and the viewpoint. Therefore, when a cluster is reached, only its elements need to be considered, in the context of the desired viewpoint. Also, notice that the viewer can be inside HC. Objects whose bounding spheres are completely behind the desired view plane are not rendered. If, however, an object is partially behind the desired view plane, only its visible samples are rendered. If a sample falls behind the desired view plane, the denominator of the rational expression (the 3-D image warping equation) that computes the sample's coordinates in the final image is negative [8]. Its sign is used in the clipping test.

4.1 Special Cases

When applying the algorithm to warp a series of objects, it is strictly correct to warp only the parts of the objects that fall in the working semi-circle. However:

1. If an object is crossed by the segment from the HCOP to e- and the angular range of the object does not overlap any other object range (Figure 8a), the object can be warped first, although its parts fall in the both semi-circles. In case the object is crossed by the segment from HCOP to e+ (Figure 8a), the object should be warped last.
2. If the angular ranges of at least two objects are crossed by segment from HCOP to e- (Figure 8b), keep applying the algorithm to this cluster recursively, and this situation will be reduced to case 1.
3. If at least one object is crossed by the segment from HCOP to e+ (Figure 8c), such an object/cluster should be warped last. Again, apply the algorithm recursively.

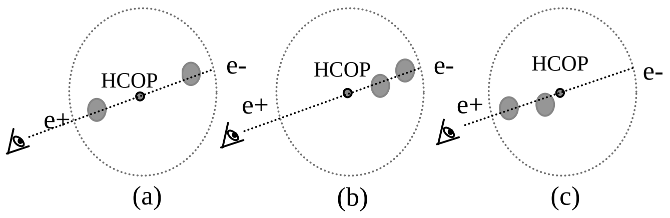


Figure 8. (a) Objects crossed by the segment connecting the viewpoint and HCOP. (b) Two objects with overlapping angular ranges crossed by the segment HCOP/e-. (c) Two overlapping angular ranges crossed by the segment HCOP/e+.

4.2 Further Considerations

In the presented algorithm, the rendering primitives are IBOs. This implies a coarse granularity and, therefore, a small number of comparisons to decide the final order. Whereas our approach can handle dynamic scenes, its main disadvantage is that it does not support object interpenetration. Such a limitation is due to the fact objects are entirely rendered according to a serial order, which precludes visibility cycles. Alternatively, dynamic BSP-trees [14] can be used to decide the serial rendering order of objects in a scene. Although we have not used dynamic BSP-trees to render IBO scenes, we suspect that both techniques may have comparable costs.

Our current implementation of the algorithm is obtained by computing the projections of the objects' bounding spheres onto a plane perpendicular to the view plane while using the procedure described in section 4. The use of bounding spheres allows for a 2-D implementation to be applied to 3-D scenes. For the cases in which two projections overlap on the plane, the ambiguity is solved by computing the projections of the conflicting objects onto planes perpendicular to the original one. If the conflict persists, for each pair of conflicting objects we compute the plane orthogonal to the vector connecting the centers of the two objects, and tangent to one of the bounding spheres. The coefficients a, b, and c of the plane are the coefficients of the orthogonal vector. d is obtained by plugging in the coordinates of the vector scaled by the radius of the first sphere. The sign of the desired view position with respect to the computed plane is then used to order the conflicting objects.

The check for angular range overlapping is implemented conservatively. For each object we compute a vector v, orthogonal to the vector from HCOP to the center of the object's bounding sphere. v's length equals the radius r of the object's bounding sphere (Figure 9a). Then, we compute vectors p_1 and p_2 (with tails at HCOP), by translating v and $-v$ by $r/2$ towards HCOP (Figure 9b). Finally, compute the angles between p_1 and p_2 and the vector from the viewpoint to HCOP (this has the same direction and orientation as the vector from HCOP to he-) (Figure 9b). The angular range comprising each object is used to check for overlappings.

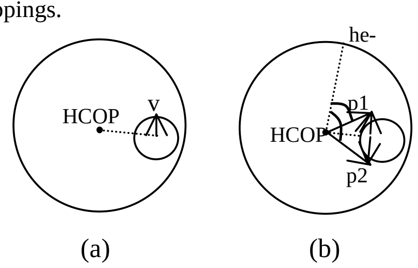


Figure 9. Angular range overlapping test. (a) v is the vector orthogonal to the segment connecting the center of the two circles. (b) The vectors p_1 and p_2 are used to compute a conservative angular range for the object's bounding circle. Angular ranges are used to check for overlapping projections.

4.3 An Approximation Algorithm

Given the restriction that spherical bounding spheres of objects do not interpenetrate, a simpler and faster approximation algorithm can be used to produce correct visibility in almost all cases. A priority list is constructed simply by sorting the objects according to the decreasing distance from the desired viewpoint to the center of each bounding sphere. Despite its simplicity, this heuristic works very well in practice. Figure 10 illustrates the concept showing two views of the same scene produced with an interactive tool used to verify the heuristic. The numbers associated with the circles are distances from the desired viewpoint to their corresponding centers. The heuristic breaks down for configurations involving spheres at highly different scales and tangent to each other. Figure 11a depicts such a configuration. Notice that this is an error-tolerant heuristic, and the rendering of the objects in the specified order can still be correct even for such configurations (Figure 11b).

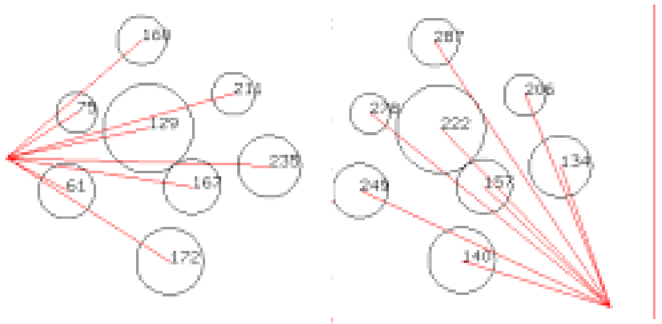


Figure 10. The use of an approximation algorithm to define a priority list. The numbers represent the distances from the desired COP to the center of the spheres. Objects are sorted by decreasing distance.

4.4 Adding Geometric Objects to a Scene

If there are no interpenetrations involving IBOs or geometric model bounding spheres, both IBOs and geometric models can be safely rendered to the same buffer. Such a situation is similar to the one involving only image-based objects. Geometric models should be rendered using a z-buffer to solve visibility among the polygons that constitute each model (*i.e.*, the z-buffer can be reset after the rendering of each geometric model).

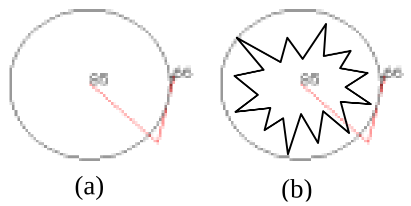


Figure 11. (a) The configuration where the heuristic breaks down: two spheres at extremely different scales and very close to each other. Although its center is closer, the smaller sphere is hidden by the bigger one. (b) This order can still produce correct results depending on the actual geometry of the objects.

5 RESULTS

We built a system prototype in C++ that implements the algorithms described for construction and rendering of IBOs. In our system, IBOs can be built using 4 different approaches: images with depth obtained from 3D Studio MAX, images with depth obtained from the OpenGL depth buffer [15], images acquired with a laser range finder, and a modified ray tracer [5] that keeps all intersections along a ray. In all examples shown (Figures 12, 13, 15, and 16), visibility was solved using our occlusion compatible order algorithm, and no anti-aliasing technique has been used. Fixed-size, totally opaque splats were used for rendering. For each object, the choice of splat size is based on the distance between the viewer and the object itself.

The old clock shown in Figure 12 was generated from 6 synthetic images rendered with 3D Studio MAX. Notice that in this case the registration process is extremely easy because we have exact camera calibration. The generalized disparity values were obtained using a plug-in. Its final representation is composed of 6 150x150 LDIs, with a total of 218,312 samples. This is equivalent to a regular depth image with 468x468 pixels.

The statue of Venus in Figure 13 was generated from an original model consisting of 90,044 polygons. Four images rendered using 3D Studio MAX were used to produce 6 150x150 LDIs, with a total of 242,488 samples. This is equivalent to a regular depth image with 493x493 pixels. Notice how this complex shape is faithfully reconstructed from a relatively small number of samples.

An Acuity Research AccuRange4000 time-of-flight laser range finder was used to create IBOs from real objects. It outputs intensity-reflected gray scale images and range maps. Figure 14 shows four views of an object acquired using a rotational platform. Each image subtends 30 degrees in both vertical and horizontal field-of-view and is 240x240 pixels in size. Specularity is a problem common to all laser range scanners [1] and, in order to reduce its effect, the specular helmet was sanded. However, specular highlights are still visible in the lateral views (Figure 14). Other major sources of error during range acquisition are the discontinuities involving the boundaries of the object and the background. In those regions, laser range finders usually receive two returns and average them, producing wrong measurements. In order to avoid this problem, a planar specular reflector oriented approximately 30 degrees with respect to the vertical was used as background. This way, as some portion of the laser beam missed the object, it was reflected away from the sensor. The background color in Figure 14 corresponds to regions where no light (zero intensity) returned to the sensor and illustrates the effectiveness of our solution. However, wrong measurements are still caused by discontinuities along the surface of the object (for instance, see the discontinuities between the face and the helmet), as well as by inaccuracies of the device. In all such cases, the error appears as noisy data (outliers).

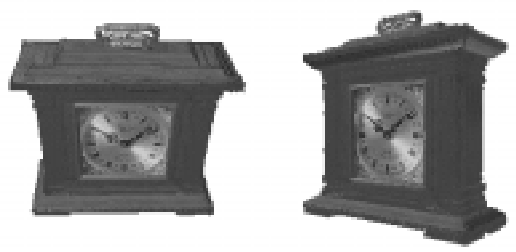


Figure 12. Views of an IBO constructed from 6 images and rendered using 2x2 splats.

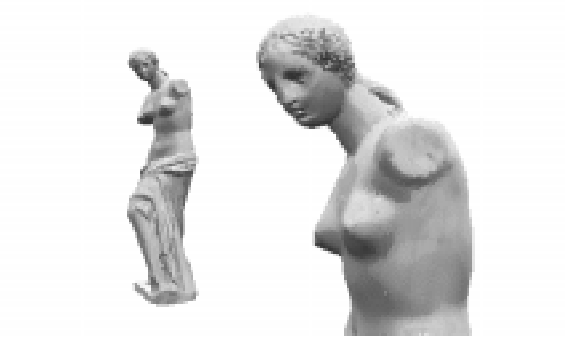


Figure 13. Views of an IBO constructed from 4 images of a highly specular statue of Venus. The left image was rendered with points, while right one was rendered using 3x3 splats.

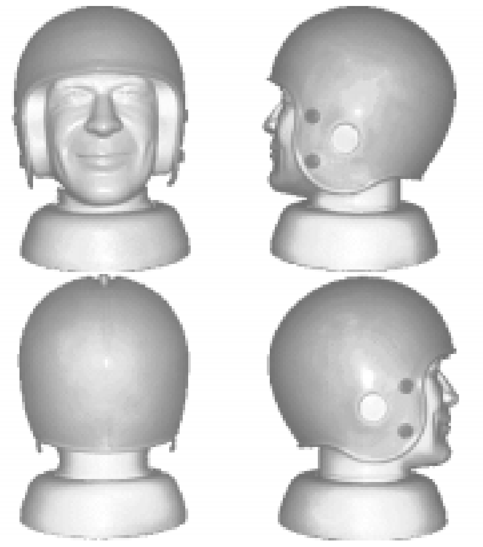


Figure 14. Reflected intensities of a real object obtained with a laser range finder. From top to bottom, left to right: 0, 90, 180, and 270 degrees, respectively. Background color represents zero intensity.

Since the reference images were acquired from virtually different COPs (the scanner COP was kept still and the platform was manually rotated), the images needed to be registered. The approach used for registration is very simple: a white pin was scanned at the center of the rotational platform. From the range data, the (x,z) coordinates of the pin with respect to the scanner were recovered (our system can provide the 3-D coordinates associated to any pixel of a range image). The samples were then rotated around the vertical axis passing through (x,z). This simple procedure led to good initial positioning that required little user intervention to achieve satisfactory (although not perfect) registration.

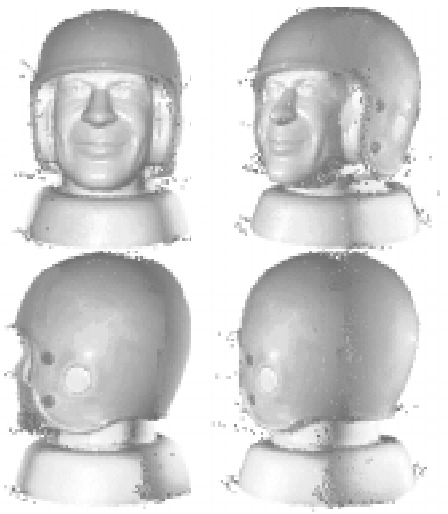


Figure 15. Views of an IBO constructed from the four range images shown in Figure 14. The visibility problem is correctly solved, but some artifacts due to noise, differences in shading in the original images, and to areas not sampled by the scanner can be noticed.

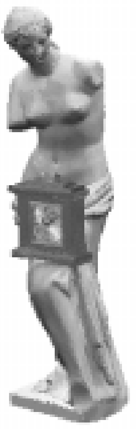


Figure 16. Venus with an old clock: inter-object visibility solved using the approximation algorithm from section 4.3 (2x2 splats).

Given the construction of the laser device, the light source is always at the eye position. This causes shading inconsistencies when multiple views are put together. Such distracting effects were reduced (but again not completely eliminated) by blending the intensity values of samples where seams were most noticeable. This was accomplished using a 3-D painting tool that is part of our system.

Figure 15 shows some views of the IBO reconstructed from the range images presented in Figure 14, and rendered using 2x2 splats. A total of 112,865 valid samples were stored in 4 150x150 and 2 100x100 LDIs, and are equivalent to a regular depth image with 336x336 pixels. Despite its small size, all major features of the original object are preserved.

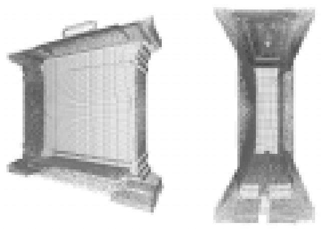


Figure 17. Close-ups of the old clock rendered as point clouds to illustrate back face culling (front and opposite side culled).

Figure 16 shows a scene with two IBOs. The visibility between the two objects was solved using the approximation list priority algorithm described in section 4.3. Notice that the two objects do not interpenetrate (although their bounding spheres do). For typical situations involving objects on flat surfaces (*e.g.*, table, floor, etc.), the plane of the surface can be explicitly used to establish the relative rendering order. Thus, for instance, if the viewer is on the same side of the plane as the objects, the flat surface should be rendered first; otherwise, last. For such cases, the model should store the coefficients of the corresponding plane equations. For applications that can constrain objects' spatial relationships, this approximation algorithm can be an attractive alternative. The accompanying videotape illustrates the use of the algorithm in a dynamic scene.

We measured the frame rates associated with the rendering of IBOs on a Pentium II PC running at 400MHz. Resampling was performed using 2x2 splats. During the measurements, all objects were completely inside the user's field of view and displayed on a 256x256 window. Table 1 summarizes the results for the IBOs shown in Figures 10, 11 and 13.

Table 1. Average frame rates and sizes of IBOs

Description	old clock	Venus	Helmet
Frames/sec.	7.13	6.26	8.29
Samples	218,312	242,488	112,865
Equiv. to a single layer image (size)	468x468	493x493	336x336

6 CONCLUSIONS AND FUTURE WORK

Performance of image-based and conventional techniques is frequently compared based on the amount of computation required to transform the modeling primitives during the rendering stage. Whereas the technique presented in this paper has comparable performance to mesh-based approaches when using conventional graphics hardware, we anticipate the development of specialized accelerators for splat-based rendering, which may make our approach more advantageous.

The use of better acquisition strategies (for instance, based on Cyberware scanners) can provide registered seamless color images to greatly improve the appearance of IBOs constructed from real objects. The use of variable splat sizes and anti-aliasing techniques can also improve the final appearance of IBOs in general.

An interesting problem is how to guarantee consistent illumination in a scene composed of multiple IBOs. Although this is not a problem for perfectly diffuse synthetic objects, inconsistent illumination can be distracting in scenes composed by real or specular objects.

Acknowledgements

We would like to thank Nick England, Anselmo Lastra, and the UNC IBR group for their assistance and support. Special thanks to Lars Nyland for his help with the laser experiments. Discussions with David McAllister resulted in the approximation algorithm. We also would like to thank the anonymous referees for their insightful comments. The old clock model was provided by Amazing 3D Graphics, and the Venus model was provided by REM Infográfica.

This work was sponsored by CNPq/Brazil – Process # 200054/95. Additional support provided by DARPA under order # E278 and NFS under grant # MIP-961. Intel generously donated equipment used in this research.

References

[1] Arman, F., Aggarwal, J. Model-Based Object Recognition in Dense Range Images – A Review. ACM Computing Surveys, Vol. 25, No. 1, March 1993. pp. 5-44.

[2] Dally, W., et al. The Delta Tree: An Object-Centered Approach to Image-Based Rendering. MIT AI Lab Technical Memo 1604, May 1996.

[3] Gortler, S., et al.. The Lumigraph. Proc. SIGGRAPH 96 (New Orleans, LA, August 4-9, 1996). In Computer Graphics Proceedings. Annual Conference Series, 1996, ACM SIGGRAPH, pp. 43-54.

[4] Grossman, J., Dally, W. Point Sample Rendering. Proceedings of the 9th Eurographics Workshop on Rendering. Vienna, Austria, June 1998, pp. 181-192.

[5] Kolb, C. Rayshad User's Guide and Reference Manual. Draft 0.4, January 10, 1992.

[6] Levoy, M., Hanrahan, P. Light Field Rendering. Proc. SIGGRAPH 96 (New Orleans, LA, August 4-9, 1996). In Computer Graphics Proceedings. Annual Conference Series, 1996, ACM SIGGRAPH, pp. 31-42.

[7] McMillan, L., Bishop, G. Plenoptic Modeling: An Image-Based Rendering System. Proc. SIGGRAPH 95 (Loas Angeles, CA, August 6-11, 1995). In Computer Graphics Proceedings. Annual Conference Series, 1995, ACM SIGGRAPH, pp. 39-46.

[8] McMillan, L. An Image-Based Approach to Three-Dimensional Computer Graphics. Ph.D. Dissertation. UNC Computer Science Technical Report TR97-013, University of North Carolina, April 1997. http://www.cs.unc.edu/~ibr/pubs/mcmillan-diss/mcmillan-diss.pdf

[9] Oliveira, M., Bishop, G. Dynamic Shading in Image-Based Rendering, UNC Computer Science Technical Report TR98-023, University of North Carolina, May, 1998. http://www.cs.unc.edu/~oliveira/TR98-023.pdf

[10] Pulli, K., et al. View-Based Rendering: Visualizing Real Objects from Scanned Range and Color Data. Proceedings of the 8th Eurographics Workshop on Rendering. St. Ettiene, France, June 1997.

[11] Rademacher, P., Bishop, G. Multiple-Center-of-Projection Images. Proc. SIGGRAPH 98 (Orlando, FL, July 19-24, 1998). In Computer Graphics Proceedings. Annual Conference Series, 1998, ACM SIGGRAPH, pp. 199-206.

[12] Schaufler, G. Per-Object Image Warping with layered Impostors. Proceedings of the 9th Eurographics Workshop on Rendering. Vienna, Austria, June 1998, pp. 151-162.

[13] Shade, J., et al. Layered Depth Images. Proc. SIGGRAPH 98 (Orlando, FL, July 19-24, 1998). In Computer Graphics Proceedings. Annual Conference Series, 1998, ACM SIGGRAPH, pp. 231-242.

[14] Torres, E.. Optimization of the Binary Space Partition Algorithm (BSP) for the Visualization of Dynamic Scenes. Proc. EUROGRAPHICS'90 (Montreux, Switzerland, September 4-7, 1990), pp. 507-518.

[15] Woo, M., et al. OpenGL Programming Guide. 2nd edition. Addison Wesley, 1997.

Appendix A

Theorem: Let B' be the base of pyramid $P_{B'}$ intersected by the segment connecting the positive epipole and its parallel projection into K. Then, warping the faces of the cube in the order (K, B, A, A', B', F), or (K, B, A', A, B', F) produce correct visibility from the desired view position.

Proof: Consider the four planes that split the cube into pyramids (Figure 4). The pyramid containing K is the only one that is always separated from the half space (defined by each of the four planes) that contains the desired view position. Thus, its samples are the only ones that cannot occlude samples from any of the other five pyramids. Therefore, K must be warped first. On the other hand, the pyramid containing F is the only one that always falls into the same half space that contains the desired view position. Thus, its samples are the only ones that can potentially occlude samples from all the other five pyramids. Therefore, F must be warped last.

After removing the pyramids containing K and F, only two of the four planes are necessary to divide the space into four disjoint subspaces, such that the pyramids containing A, A', B, and B' all fall into different subspaces. The pyramid containing B is the only one that is always separated from the half space (defined by each of the two remaining planes) that contains the desired view position. Thus, its samples cannot occlude samples from any of the other three pyramids. Therefore, B should be warped second. On the other hand, the pyramid containing B' is the only one that always falls into the same half space that contains the desired view position. Its samples can potentially occlude samples from pyramids containing A and A'. Therefore, B' must be warped fifth.

Two opposite pyramids not containing the epipoles cannot occlude each other, since there is a plane passing through the desired COP and the apices of the pyramids that separates them into different half spaces (actually, there are infinitely many such planes). Thus, two opposite pyramids project into distinct portions of the desired view plane, and therefore cannot occlude each other. Thus, it is safe to warp either A third and A' fourth or A' third and A fourth.

Notice that at first glance it seems that A, A', B, and B' can be warped in any order, given that K is warped first, and F is warped last. However, this is true only when the positive epipole falls exactly at the center of face F.

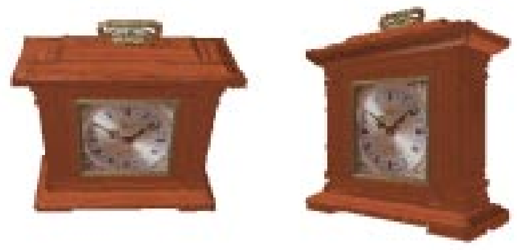


Plate 1 (Figure 12) Views of an image-based object (IBO) rendered using 2x2 splats.

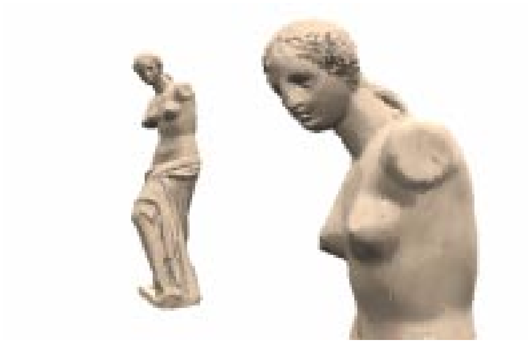


Plate 2 (Figure 13) Views of an IBO constructed from 4 images of a highly specular statue of Venus. The left image was rendered with points, while right one was rendered using 3x3 splats.

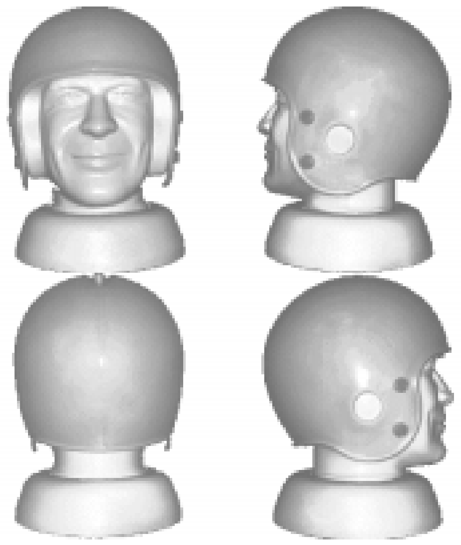


Plate 3 (Figure 14) Reflected intensities of a real object obtained with a laser range finder. From top to bottom, left to right: 0, 90, 180, and 270 degrees, respectively. Background color represents zero intensity.

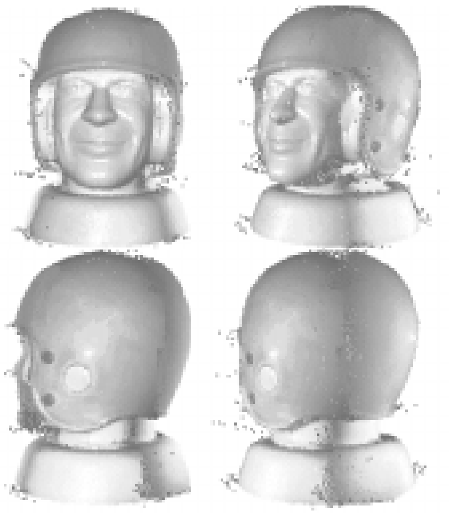


Plate 4 (Figure 15) Views of an IBO constructed from the four range images shown in plate 3. The visibility problem is correctly solved, but some artifacts due to noise, differences in shading in the original images, and to areas not sampled by the scanner can be noticed.

Plate 5 (Figure 16) Venus with the old clock: scene rendered using the approximation algorithm from section 4.3 (2x2 splats).

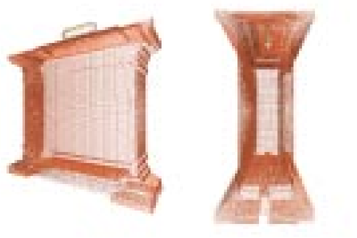


Plate 6 (Figure 17). Close-ups of the old clock rendered as point clouds to illustrate back face culling (front and opposite side culled).