Automatic Camera Placement for Image-Based Modeling

Shachar Fleishman Daniel Cohen-Or
Computer Science Department
Tel-Aviv University, Ramat-Aviv 69978, Israel
{shacharf,daniel}@math.tau.ac.il

Dani Lischinski
Institute of Computer Science
The Hebrew University, Jerusalem 91904, Israel
danix@cs.huji.ac.il

Abstract

We present an automatic camera placement method for generating image-based models from scenes with known geometry. Our method first approximately determines the set of surfaces visible from a given viewing area and then selects a small set of appropriate camera positions to sample the scene from. We define a quality measure for a surface as seen, or covered, from the given viewing area. Along with each camera position, we store the set of surfaces which are best covered by this camera. Next, one reference view is generated from each camera position by rendering the scene. Pixels in each reference view that do not belong to the selected set of polygons are masked out.

The image-based model generated by our method, covers every visible surface only once, associating it with a camera position from which it is covered with quality that exceeds a user-specified quality threshold. The result is a compact non-redundant image-based model with controlled quality.

The problem of covering every visible surface with a minimum number of cameras (guards) can be regarded as an extension to the well-known Art Gallery Problem. However, since the 3D polygonal model is textured, the camera-polygon visibility relation is not binary; instead, it has a weight — the quality of the polygon's coverage.

1. Introduction

Image-based modeling and rendering (IBMR) is a new paradigm in computer graphics, which has gained considerable popularity and has been the subject of much recent research in the field. In this paradigm, a 3D scene is modeled as a collection of *reference images*, rather than a set of conventional geometric primitives [2, 12]. Novel views of a scene can then be synthesized from this collection of images using a variety of interpolation and re-projection techniques [2, 6, 10, 12, 17]. IBMR possesses several important advantages over traditional modeling and rendering: (i) the images can be of real world scenes, thus making the task of modeling such scenes much easier; (ii) image-based rendering algorithms are typically inexpensive and can be carried out on general purpose computers; (iii) the rendering time is typically independent of the geometrical and physical complexity of the scene being rendered. These last two properties make IBMR extremely useful for rapid rendering of complex synthetic 3D scenes [2, 10, 11, 14, 15, 16].

In their landmark paper, McMillan and Bishop [12] have placed image-based rendering in the framework of sampling and reconstruction of the plenoptic function. Most of the work since has been done on reconstruction algorithms (efficient warping, splatting, etc.), but the important problem of properly sampling the plenoptic function has remained largely unanswered. To illustrate the problem, consider the following example. Suppose that we would like to compute an image-based model of a virtual museum, so that users will be able to walk through it at interactive rates using image-based rendering. In order to generate a set of reference images for this scene we must decide where to place the cameras from which these images will be rendered. Ideally, the camera placement problem must take the following considerations into account:

1. Every polygon in the scene that might be seen by a user in the course of a walkthrough must be visible, or *covered*, in at least one reference image.

2. Every covered surface should be sampled at a sufficiently high rate (cover sufficiently many pixels in the reference images). Otherwise, it might appear too blurry in the course of a walkthrough. This is particularly important for textured polygons, such as the pictures on the walls of the museum. We refer to the rate at which a polygon is sampled in an image-based model as its *coverage quality*. (A more precise definition will be given in Section 3).

To our knowledge, there is currently no method for automatic placement of reference views in a general 3D scene in a manner that would provide adequate coverage: i.e., every surface of interest is covered, with some lower bound

on the coverage quality. For lack of a better alternative, *ad hoc* camera placement methods are typically employed. For example, in McMillan's plenoptic modeling system [12], cylindrical reference views are placed on a regular grid in the scene. Such naive placement of reference views cannot guarantee that all surfaces of interest are adequately covered. Additionally, the resulting image-based model is highly redundant, as the same surface might be represented in many (perhaps even all) reference views. One attempt to address the camera placement problem is described by Stürzlinger [18]. However, his method does not address the important issue of coverage quality. In another work, Grossman *et al.* [7] propose to compute 32 orthographic projections of an object, and then use an iterative greedy algorithm to find a set of blocks that provides adequate sampling of the object. Grossman's approach is object-centered — the goal is to cover an object, in contrast to our approach that is view-centered — the goal is to provide a coverage of an entire scene from a set of views. Furthermore, using 32 predetermined orthographic views may still miss important features.

This paper describes a first attempt to provide a practical solution to automatic camera placement with adequate coverage. At this point, we should note that adequate coverage is only possible if the space from which a user may view the scene is restricted and empty of scene objects. Otherwise, the user is able to approach a surface in the scene arbitrarily close, and no *a priori* sampling rate can guarantee a lower bound on the coverage quality. Thus, we assume that the viewpoint may move only inside a predefined empty *viewing area* or *walking zone*. We define the walking zone by a set of boundary polygons. Figure 1, illustrates a scene and a walking zone. We also assume an upper bound on the resolution at which images of the scene are to be generated. These two assumptions define an upper bound for the sampling rate of each surface in the scene. Finally, we assume that the scene consists of ideal diffuse (Lambertian) surfaces with texture.

The contribution of this paper is a practical method for generating image-based models of synthetic 3D scenes. The generated model covers most (and in many cases all) of the surfaces that may be visible from the walking zone. Every covered surface is represented exactly once, and the coverage quality exceeds a user-specified threshold.

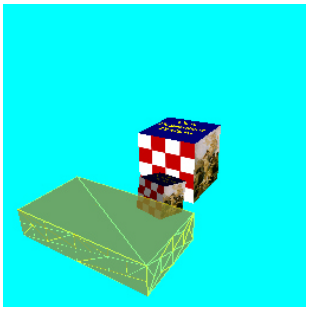
Our method determines a small, though not necessarily minimal, set of cameras (the term *camera* refers to all of the relevant viewing parameters, such as position, orientation, field of view, resolution, etc.) With each camera we associate an exclusive set of surfaces covered in the corresponding view. To generate an image-based model, we traverse this set of cameras, and render the corresponding reference images. In each image we mask out pixels which do not belong to surfaces covered in this view. Figure 2 shows the masked reference images produced by our algorithm for a simple test scene.

Figure 1. A general view of a test scene. The walking zone is the semi-transparent box in the middle of the room.

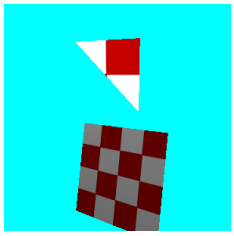
The resulting set of *masked reference images* constitutes an image-based model of the scene. Taking our method to an extreme, we could have as many masked reference images as visible polygons — for each polygon we could find the best location in the walking zone to view it from. In practice, our solution does not strive to go that far. On the contrary, we would like to minimize the total number of reference views, in order to increase the image coherence of our representation, thus allowing warping to be performed more efficiently.

Our automatic camera placement can be also useful for placing cameras when capturing real world scenes, provided that a (possibly approximate) 3D geometric model of the scene is available. In this case, our method will produce a small number of camera positions from which photographs should be taken in order to construct an image-based model of the scene [6].

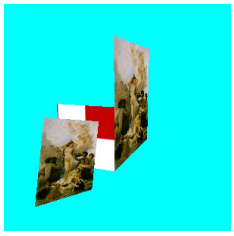
The proposed method is also applicable for selecting the best views along a known walkthrough path. In the synthetic video compression technique of Cohen-Or *et al.*[5], a video sequence is reconstructed by rendering the model with view-dependent texture maps. These texture maps are extracted from selected precomputed reference views along the walkthrough path. A successful selection of these reference views will minimize the amortization of the view-dependent textures along the sequence, thus yielding better compression.



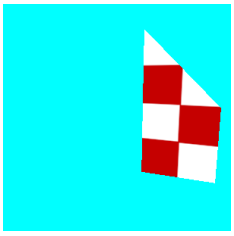
A simple 3D scene with a walking zone.



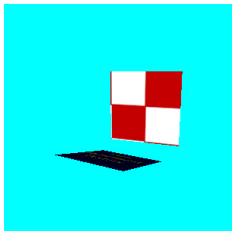
(c1)



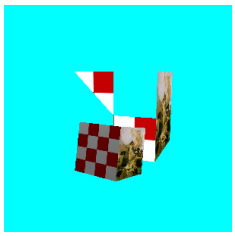
(c2)



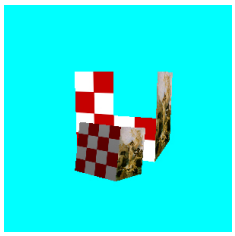
(c3)



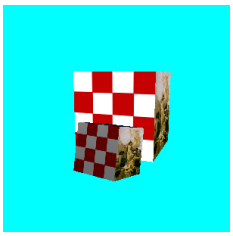
(c4)



(c1+c2)



(c1+c2+c3)



(c1+c2+c3+c4)

Figure 2. Masked reference views. Images c1–c4 show four reference views of a simple 3D scene. The cyan areas are masked out. The bottom row of images illustrates the cumulative coverage of the scene by these reference views.

2. Overview

Ideally, given a polygonal 3D scene and a walking zone, we would like to determine precisely which parts of the scene are visible from at least one viewpoint inside the walking zone, and then select the minimum number of cameras covering every visible part. The first task requires computing the antipenumbra [19] of the walking zone, while the second is the 3D version of the well-known Art Gallery Problem, known to be NP-hard [13]. Thus, in order to make our method practical, we must resort to approximations. Our method first approximates the set of scene polygons visible from the viewing zone, and then employs a greedy algorithm to select a small number of camera positions that together cover every polygon in the above set. For every camera, we store a list of polygons whose coverage quality from that view is above a given threshold. These lists are disjoint, and their union is the set of all visible polygons.

Our method is based on populating the walking zone with a large number of camera positions and then selecting a small subset of camera positions from the above set. To reduce the number of camera positions, we place them only on the boundary of the walking zone. Since the walking zone is empty, it follows that for every point p in the scene that is visible from some point v inside the walking zone, p can also be seen from the intersection point between the line (v, p) and the boundary of the walking zone. This argument can be extended from points to polygons, provided that they are sufficiently small (otherwise, cases such as the one shown in Figure 3(a) may arise).

The algorithm performs the following steps:

1. **Tessellate the boundary:** The boundary of the walking zone is tessellated into small patches.

2. **Subdivide polygons:** Scene polygons are subdivided into smaller ones in order to reduce the likelihood of

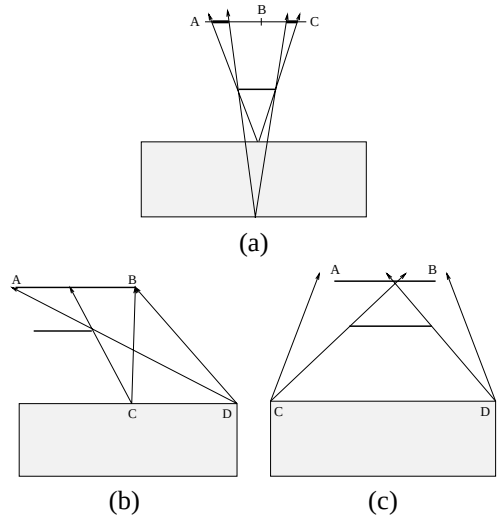


Figure 3. Partial visibility problems: (a) Polygon AC has two disjoint visible regions. The polygon must be subdivided into AB and BC. (b) Polygon AB is only partially visible from a nearby position C, but is completely visible from a farther position D. (c) Polygon AB is not fully visible from a single viewpoint within the walking zone, but is fully covered if views C and D are combined. Polygon AB must be subdivided.

the visibility problems illustrated in Figure 3.

3. **Compute visibility:** For each patch, compute the hemispherical image seen from the center of the patch (Section 4). The set union of polygons visible in all these images is our approximation to the set of visible polygons.

4. **Compute best quality:** Compute a per polygon value that measures the best quality a polygon can achieve from a camera placed within the walking zone (Sections 3 and 4).

5. **Select camera positions:** Select a small subset of camera positions from the above database that meets our goals for a fully covered scene with adequate coverage quality (Section 5).

3. The coverage quality

In this section, we define a quality measure for a polygon in an image. We define $V(p_i, I_j)$, the *value* of a given polygon p_i with respect to a given camera C_j, as the number of pixels that p_i contributes to I_j, the hemispherical image of the scene at C_j. This value depends on the solid angle of the polygon, as seen from C_j, the visibility of p_i from C_j, and the resolution of the image I_j. In fact, it is proportional to the point-to-polygon form-factor [4] between C_j and p_j. In our context, a high value means that the texture of the polygon is sampled at a high rate in image I_j.

Given a walking zone in a polygonal 3D scene, we can define the *best value* of a polygon p as seen from the zone as $BV(p) = \max_{j \in all\ camera\ positions}(V(p, I_j))$. The best value of a polygon as seen from the walking zone defines the sampling rate that is required to sample the polygon from the walking zone, such that the polygon is neither oversampled nor undersampled.

Now, we can define the *coverage quality* of a polygon p in an image I taken from the zone as $Q(p, I) = \frac{V(p,I)}{BV(p)}$. This is a number in the range $[0, 1]$ that measures how well the polygon p is covered in image I.

The quality measure defined above was developed with fully visible polygons in mind, and it is not well suited for ranking the coverage of partially occluded polygons. For example, a polygon may have a higher coverage quality in a partially occluded view than in a more distant view from which it is completely visible (Figure 3(b)). Some polygons may be fully covered only by a combination of more than one view (Figure 3(c)), so picking a single best view for such a polygon will lead to a visibility gap in the resulting image-based model. These problems are more common when scene polygons are large with respect to their distance from the walking zone, and tend to disappear if the polygons are sufficiently small. Thus, subdividing the input polygons before processing them alleviates most of the partial visibility problems. However, one must take care not to subdivide polygons too much, since polygons covering only a small number of pixels in the reference images are more prone to discretization errors.

Another limitation of the quality measure is related to our assumption that the scene is ideal Lambertian: coverage quality only refers to diffusely shaded texture, without accounting for view-dependent shading.

4. Computing visibility

In order to place cameras, we need to find the subset of polygons in the scene that can be seen from the walking zone. For each visible polygon, we need to find the areas inside the walking zone from which that polygon is visible and to compute the best value of every visible polygon.

We approximate the above computation using the following approach. The shell of the walking zone is tessellated into small patches, and the center of each patch is chosen as a potential camera position. From each position, we compute a hemispherical image of the scene. Each pixel in this image contains the unique ID of the scene polygon

visible through that pixel. By counting the number of pixels covered by the projection of each polygon in each of these images, we obtain the value $V(p_i, I_j)$ of every polygon p_i in each image I_j. Polygons whose value is 0 in all images are considered invisible from the walking zone. The approximate set of visible polygons is the union of visible polygons in all of the images I_j. We approximate the best value of a polygon as the maximum value of the polygon over all images.

In practice, we use 3D rendering hardware to speed up the process of visibility computation. Rather than projecting the scene onto the hemisphere around each camera position, we render $(\frac{180}{fov})^2$ planar images that approximate the hemisphere (*fov* is a user-specified field-of-view). Every camera position is saved along with a list of polygons visible from it and their corresponding values.

The algorithm is stated in detail below:

1. Assign a unique ID for each polygon in the scene.
2. For every polygon with ID $= i$, let $BV(p_i) = 0$.
3. Let $camera\ database = \emptyset$
4. For each walking zone shell patch j do:
   (a) Divide the hemisphere directed outside of the shell to $K = (\frac{180}{fov})^2$ sections.
   (b) For each section $[1..K]$ do:
      i. Set a camera at the center of patch j pointing at the center of the current section.
      ii. Render the scene into an item buffer IB.
      iii. Traverse IB and calculate the values of polygons that are visible in this view.
      iv. Update the value of every visible polygon i: $BV(p_i) = \max(BV(p_i), V(p_i, \text{IB}))$
      v. Insert $(camera\,position, visible\,polygons)$ to $cameras\ database$

There are cases where a polygon has a higher value in a partially occluded view C_j than in another view C_k from which it is completely visible (Figure 3(b)). In such a case C_k should be preferred over C_j in order to avoid visibility gaps in the coverage of the scene. Thus, the algorithm above can be improved by adding the following phase:

5. For every polygon p_i that is visible from C_j
   (a) If (partially_visible(p_i, C_j) and (p_i is fully visible from some other viewpoint))
      i. remove p_i of the list of polygons visible from C_j.
      ii. Update $BV(p_i)$ to reflect the removal of p_i from C_j.

5. Selecting camera positions

At this stage, for each polygon in the scene, we can tell the optimal rate (up to our approximation error) that is required to sample the polygon for viewing it from anywhere inside the walking zone. We also know which camera position samples the polygon at the optimal rate. From this information we can generate an image-based model simply by sampling each polygon individually from the camera which covers it the best. Such an approach would result in a very large number of reference images, and hence suffers from two drawbacks: (i) if we apply this algorithm to select camera locations for taking pictures of a real scene, we would end up with a tremendous amount of pictures to take; (ii) we cannot take advantage of the speed of incremental image warping when rendering the image-based model, because each reference image will contain too few pixels.

Our approach, instead, is to compromise: rather than insisting that each polygon in the scene is covered with optimal quality, we are willing to allow lower quality coverage so long as the quality exceeds some user-specified minimum Q. Our goal now is to select a small set of cameras such that all visible polygons are covered with quality $\geq Q$.

To generate the smaller set of camera positions, we use a greedy approach. For each camera in the database constructed earlier we compute a list of polygons that are adequately covered by that camera. A polygon is considered adequately covered by a camera, if its quality in the corresponding image is larger than the user-specified quality threshold Q. Starting with an empty set of cameras C, at every iteration we add the highest ranked camera from the database. The *rank* of a camera is defined as the number of adequately covered polygons that it adds to those already covered by previously selected cameras. Cameras are added to C until all of the visible scene polygons are adequately covered by the union of the selected cameras.

It often happens that camera C_1, selected earlier by the greedy algorithm, covers polygon p with lower quality than some other camera C_2, selected later in the process. In such a case, when we add C_2 to the selected camera set C, we remove p from the set of polygons covered by C_1 and add it to the set of C_2. Such changes have no effect on the resulting number of selected cameras, but they result in better coverage quality.

The camera selection algorithm described above terminates only when the entire visible set of polygons has been covered. An alternative problem that sometimes needs to be solved is selecting the k camera positions that cover as much of the scene as possible. We could solve this problem using the same greedy algorithm, and stop when $|C| = k$. Since it is probable that not all polygons can be covered by k cameras, it makes sense to favor polygons that are most likely to be seen. For that purpose, we change the rank-

ing of candidate cameras in the following way: for every polygon p, we count the number of cameras that cover it $|p|$. Candidate cameras are ranked according to the sum $\Sigma_{p \in covered(C)} |p|$.

6. Creating an image-based model

The selected set of camera positions and their associated covered polygon sets serve as a basis for generating an image-based model of the scene. The set of camera positions is traversed, and the scene is rendered from each position using the stored camera parameters to generate an image. Next, we use the set of polygons covered by each camera to extract the pixels covered by their projections in the corresponding image. All of the other pixels in the image are masked out. The remaining unmasked pixels form an image-based representation of the scene. This representation is non-redundant and thus it is faster to render or to transmit over the network. Yet, it guarantees a predefined coverage quality and no (or very few) visibility gaps.

Our method samples nearby surfaces in a single image when possible. In practice the results obtained from our system are a small set of images that are mostly filled and cover most of the scene and another set of images, each sampling a small number of the remaining surfaces. This behavior of the method, leads us to create a model that consists of two parts. The first part is a set of images with depth, to which incremental 3D warping can be applied in the rendering phase. The second part is a sparse set of pixels that are stored in a compact data structure.

Other types of image-based models can also be generated from our representation. For example, our image-based model is easily converted to a Layered Depth Image (LDI) representation [17] of the scene by warping all of the unmasked pixels in our representation to a common view. In order to maintain the coverage quality the LDI data structure should support multiple resolutions. An appropriate data structure is described by Chang *et al.*[1]. In this way, the LDI representation of the scene is guaranteed to be of high quality, and no visibility gaps will be created while the camera position remains inside the walking zone. Yet the LDI will be free of redundant samples, therefore rendering will be faster.

7. Results

The algorithms described in this paper were implemented on a Pentium-based PC. The system reads VRML 1.0 files, computes the set of visible polygons, selects a set of camera positions, and generates an image-based model of the scene using these camera positions. From the image-based model, novel views of the scene are rendered by forward warping of the unmasked pixels in the computed reference views. In order to demonstrate the coverage quality resulting from our approach, no hole-filling, filtering, or other anti-aliasing methods are applied to the projected samples. Figures 4 and 5 show a comparison of an image-based model that was generated based on our algorithm with an image-based model that was generated by sampling the scene with reference views placed on a regular grid. As can be seen in Figure 4(a) and 5(a), there are some small holes (in cyan) in the novel view generated from our image-based model. These holes are due to discretization problems and they could be easily eliminated by using micro-polygons [10] or splatting [17]. In contrast, the large holes in Figure 4(b) and 5(b) are due to visibility gaps, that is, surfaces not covered by the reference views placed on a regular grid.

The coverage quality of our image-based model is visualized in Figure 4(c) and 5(c), where each polygon is pseudo-colored according to its quality. Colors change from light green (high coverage quality) to dark green, dark red and bright red (not covered at all). Note that the quality of the regularly sampled images in the (d) columns was computed with respect to the best polygon values computed by our algorithm.

For these models the number of samples/pixels in the regular grid model is larger by a factor of two than the number of unmasked pixels in our model. This factor is scene dependent, however, and as the occlusion complexity of the scene grows, the factor tends to grow, since the regular sampling must be denser in order to cover the scene properly. On the other hand, the number of samples in the masked views is a function of the visible surfaces only.

8. Discussion and future work

We have presented a method for generating image-based models from textured polygonal scenes. The effectiveness of the new method is threefold:

- the set of all polygons visible from the walking zone is well approximated;
- the coverage quality of the image-based model is user-controlled;
- the representation is compact.

The masked reference views cover each visible surface only once with a user-defined minimal quality. Since the masked views are generated by taking pictures from within the walking zone, the surfaces are neither oversampled nor undersampled. In particular, faraway surfaces are sampled at the proper lower resolution. Thus, the result is a compact image-based model with controlled quality.

To generate the image-based model, we produce a set of camera positions from a large database of camera positions.

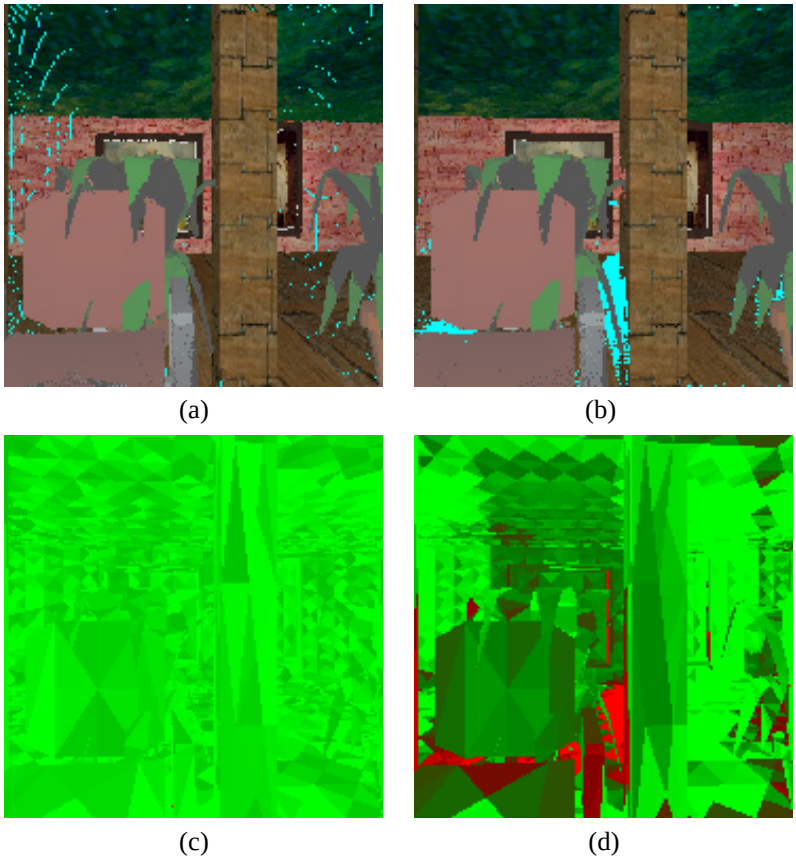


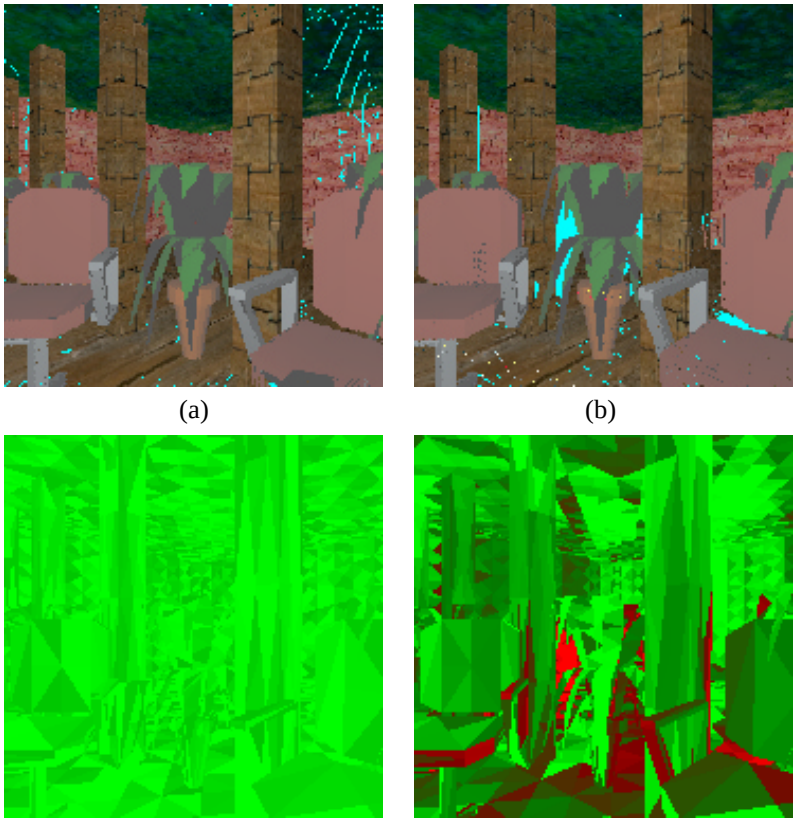
Figure 4. A view of a test model. (a) Rendered from the image-based model produced using our approach. (b) Rendered from an image-based model with regular grid camera placement. (c) Coverage quality in our image-based model: color varies from green (high quality) to red (not covered at all). (d) Coverage quality with regular grid camera placement.

The problem of selecting the minimal subset of cameras that covers every visible polygon is equivalent to the set-cover problem, where camera positions form one set of vertices, visible polygons form the second set, and the visibility information is represented by edges connecting between the set of camera positions and the polygons. Greedy heuristics like the one we use to select the set of covering cameras, produce a solution that is within a factor of $1 + \log(d)$ of the optimal solution, where d is the maximum number of cameras that sees a single polygon[3, 9].

In this paper we addressed the problem of placing cameras to cover a given 3D model. The problem can be regarded as an extension to the 2D visibility problem known as the Art Gallery Problem [13]. In our case, the 3D polygonal model is textured and the goal is to generate an effective image-based model, rather than to solve a pure visibility problem. Assuming the model has no textures and is not shaded, our method provides an approximate solution to the 3D art gallery problem.

We have assumed that all surfaces in the scene are ideal diffuse. Thus, a surface can be sampled from any direction, so long as the coverage quality exceeds the desired minimum. In order to extend our approach to non-diffuse reflectors, the material properties of the surface and the direction from which it is viewed should be incorporated into the quality metric. Shiny surfaces may have to be sampled from multiple camera locations. Alternatively, a post-warping view-dependent shading stage may have to be applied whenever a novel view is rendered from the image-based model.

Our current visibility algorithm is fairly naive; we subdivide the polygons in the scene to small polygons and let the user define the density of cameras that are placed on the boundary of the walking zone. The next logical step would be to switch to an automatic adaptive refinement strategy, resembling the hierarchical radiosity algorithm [8].



(a) (b) (c) (d)

Figure 5. Another view of the same model.

Acknowledgments

This work was supported by a grant from the Israeli Ministry of Science.

References

[1] C.-F. Chang, G. Bishop, and A. Lastra. LDI tree: A hierarchical representation for image-based rendering. In A. Rockwood, editor, *Computer Graphics* Proceedings, Annual Conference Series. ACM SIGGRAPH, Addison Wesley, Aug. 1999.

[2] S. E. Chen and L. Williams. View interpolation for image synthesis. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, Aug. 1993.

[3] V. Chvátal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4:233–235, 1979.

[4] M. F. Cohen and J. R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, San Diego, CA, 1993.

[5] D. Cohen-Or, Y. Mann, and S. Fleishman. Deep compression for streaming texture intensive animation. In A. Rockwood, editor, *Computer Graphics* Proceedings, Annual Conference Series. ACM SIGGRAPH, Addison Wesley, Aug. 1999.

[6] P. E. Debevec, Y. Yu, and G. D. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In G. Drettakis and N. Max, editors, *Rendering Techniques '98*, pages 105–116. Springer-Verlag, 1998.

[7] J. P. Grossman and W. J. Dally. Point sample rendering. In G. Drettakis and N. Max, editors, *Rendering Techniques '98*, pages 181–192. Springer-Verlag, 1998.

[8] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.

[9] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9:256–278, 1974.

[10] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3D warping. In M. Cohen and D. Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 7–16. ACM SIGGRAPH, Apr. 1997.

[11] N. Max. Hierarchical rendering of trees from precomputed multi-layer Z-buffers. In X. Pueyo and P. Schröder, editors, *Rendering Techniques '96*, pages 165–174. Springer-Verlag, 1996.

[12] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In R. Cook, editor, *SIGGRAPH 95*

Conference Proceedings, Annual Conference Series, pages 39–46. ACM SIGGRAPH, Addison Wesley, Aug. 1995.

[13] J. O'Rourke. *Art Gallery Theorems and Algorithms*. The International Series of Monographs on Computer Science. Oxford University Press, New York, NY, 1987.

[14] G. Schaufler. Per-object image warping with layered impostors. In G. Drettakis and N. Max, editors, *Rendering Techniques '98*, pages 145–156. Springer-Verlag, 1998.

[15] G. Schaufler and W. Stürzlinger. A three dimensional image cache for virtual reality. *Computer Graphics Forum*, 15(3):227–236, Aug. 1996.

[16] J. Shade, D. Lischinski, D. Salesin, T. DeRose, and J. Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. In H. Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 75–82. ACM SIGGRAPH, Addison Wesley, Aug. 1996.

[17] J. W. Shade, S. J. Gortler, L. He, and R. Szeliski. Layered depth images. In M. Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 231–242. ACM SIGGRAPH, Addison Wesley, July 1998.

[18] W. Stürzlinger. Imaging all visible surfaces. In *Proceedings of Graphics Interface '99*, pages 115–122, June 1999.

[19] S. J. Teller. Computing the antipenumbra of an area light source. *Computer Graphics*, 26(2):139–148, July 1992.