Image-based Object Representation by Layered Impostors

Schaufler Gernot
GUP, Johannes Kepler Universität Linz
Altenbergerstr. 69, A - 4040 Linz, Austria / EUROPE
gs@gup.uni-linz.ac.at

ABSTRACT

Image based object representations become increasingly popular because they offer the advantage of being unaffected by the object's complexity both in terms of rendering time and memory consumption. This paper introduces such a representation which stands out by the low number of images required, the projection method and the warping method applied to generate and use the images. One of the available depth-augmented images is chosen and warped to match the current point of view. Both image generation and image warping are supported on currently available graphics accelerators.

KEYWORDS

image-based modelling & rendering, warping, projections.

1 INTRODUCTION

A lot of computer graphics research has gone into object representations consisting of polygonal patches. As the aim is to create images of these objects, the patches must be rendered one by one. Recently, this approach has been put into question and image-based modelling has been proposed as an alternative [15] because the difficult geometric modelling step is avoided. In particular for capturing real world objects with high realism the image-based approach has already proven more straightforward and easier to automate [3][6][7][9][16].
Apart from the advantages during the modelling step, image-based modelling in combination with image-based rendering is attractive. Rendering algorithms can be constructed which are independent in complexity of the depicted object's complexity. As the number of polygons in geometric models is already approaching the pixel counts in the final image, such algorithms can be expected to outperform traditional polygonal rendering techniques. Especially objects with a fuzzy surface [16] or with complex reflectance characteristics [7][9] are difficult to model with polygons and traditional material descriptions and even more difficult to render with realistic shading with traditional rendering algorithms.
Despite of these attractive properties a number of drawbacks are still hindering widespread use of image-based techniques. These include large storage requirements and a lack of guarantees for the completeness of the representation. Previous techniques use a large number of images for representing one object resulting in the mentioned memory overhead. However, even with this large number of images there is no guarantee that the complete surface of the object has been captured. Holes can appear in the final images, because parts of the object should be visible which were not visible in any of the images used to represent the object. So far these problems have been addressed by compression schemes [9] and restrictions in the allowable viewing positions for the final images [2][3].
Recently, image warping has been used to decrease the number of images required to fully represent an object. Possible approaches include mesh approximations to the object's shape in image space [5][13][16] or full image warping on a pixel-by-pixel basis [2][6]. Holes in the final image have also been avoided by using more than one depth layer in the images of the object representation [8][11]. Both image warping and multiple z-layers suffer from the fact that they are not readily supported by current graphics accelerators which results in a performance penalty when compared to (hardware-assisted) polygonal rendering.
In the following previous work section image-based object representation approaches will be compared by how they converge to the original object and if such a convergence is indeed possible. Polygonal models can always be made to converge to the real surface by making the polygonal facets sufficiently small. In contrast, image-based representations have several parameters which need to be varied to obtain increasingly better approximations of the real object: image resolution, the number of images stored per object and the number of geometric primitives in hybrid representation schemes. Section 3 introduces the new image-based object representation scheme based on layered impostors. Section 4 gives results obtained from its implementation and section 5 concludes with a summary and directions for future work.

2 PREVIOUS WORK

In image-based modelling both object representations and whole scene representations have been explored. The two differ in that object representations try to capture the "outside-in" appearance of an object and are often limited to viewpoints at a certain distance from the object, whereas scene representations capture the "inside-out" view of the scene from one viewpoint or a set of viewpoints within the scene.

Scene Representations. Quicktime VR [3] supports both representations though in quite different ways: For scenes a portion of an environment map is warped into a perspective projection. For objects the closest available view of the object is displayed out of a "navigable movie". Plenoptic modelling [15] provides scene modelling from cylindrical environment maps. Static impostors [10] apply the navigable movie concept both per-object and per object group in a hierarchical scene graph.
In order to increase the applicability of images in image-based representations, they can be warped to better match new viewpoints. Chen et al. [2] use quad-trees in image space to exploit the coherence in this warp. Sillion et al. [13] and Darsa et al. [5] approximate the warp with polygonal meshes.

Object representations. Object representations have been used both in high-quality rendering [11] as well as in real-time

rendering [8]. Both Max and Gortler et al. avoid holes in the final images by storing more than one depth layer per pixel. While these images cannot be generated directly with conventional renderers they capture the whole geometry of the object from a single point of view.
When several images of the same object are available which were taken from different viewpoints, integrating these images into a single object representation requires careful registration. This process has been somewhat relaxed by Pulli et al. [16] who resort to soft z-buffering to generate the final image. Debevec et al. [6] tackle image registration in an interactive system. The images are projected on a crude geometric model of the object. By applying view-dependent texture mapping, high quality rendering is possible even with these crude models.
The Lightfield [9] and Lumigraph [7] approaches sample the complete plenoptic function and use interpolation to derive new images from these samples. They are applicable both to object representation and scene representation but suffer from high memory requirements. The authors of the lightfield representation have also shown how to exploit texture-mapping hardware for fast rendering.
Delta trees [4] are an effective compression technique to combine multiple images of an image-based object representation into a single data structure. As more images are inserted into the data structure only those pixels are retained which show regions of the object's surface not sufficiently sampled so far. Thereby, compression ratios of around 10:1 are achievable in comparison to straightforward image storage.
Table 1 summarizes and compares the various modelling approaches mentioned based on the modelling dimensions used: image resolution, number of images and additional geometry. In general, convergence against the true object or scene only occurs, if the images in the representation are warped. Otherwise, either an infinite number of images is required, only a finite set of viewpoints is possible or the perspective distortion in the final image is wrong. Switching between too few images will result in "popping" effects.
The maximum required image resolution is determined by the closest desired zoom-in on the objects. Techniques which apply an approximate geometric model to affect the image warp also require the geometry to converge against the true object's surface thereby losing the complexity independence of image-based rendering somewhat.

Approach	Image Resolution	Number of Images	Auxiliary Geometry	Warping, Convergence
Quicktime VR for objects [3]	X	X		NO
Quicktime VR for scenes [3]	X			NO
Static Impostors [10]	X	X		NO
View Interpolation [2]	X	X		YES
Plenoptic Modelling [15]	X			YES
Impostor Manipulation [13]	X	X	X	YES
Tree Rendering [11]	X	X		YES
Depth-layered Images [8]	X	X		YES
View-based Rendering [16]	X	X	X	YES
Hybrid Modelling [6]	X	X	X	YES
Lightfield [9] / Lumigraph [7]	X	X		YES
Delta Tree [4]	X	X		YES

Table 1: Comparison of image-based modelling approaches by modelling dimensions

In all these approaches it remains unclear if all parts of the surface of the object have been sufficiently sampled to allow the reconstruction of images from arbitrary viewpoints. In particular surface portions on concave objects may not have been observed in any of the images except if multiple depth layers per image are used. However, even in the presence of multiple depth layers surface parts almost parallel to the projection rays may not have been sufficiently sampled.
The approach presented in this paper does not completely solve the question of completeness of an object's representation but improves each image's relevance for reconstructing new images. The new projection scheme used to generate the images in the representation has the additional advantage that both image generation and image warping are supported on current graphics accelerators.

3 Object Representation with Layered Impostors

This chapter describes how an image-based object representation can be constructed using layered impostors. A number of depth-augmented images are kept per object and layered impostors are used to warp one of these images to match any new point of view.

3.1 Choosing a projection

Different types of projections have been used for storing images of objects. Environment maps (cubical, cylindrical or spherical projections) capture most of an object's surface. However, curved projections are difficult to generate using current graphics hardware which is restricted to orthogonal or perspective projections. With a typical perspective projection and a field of view of 90 degrees six projections are necessary to capture a cubical environment map.

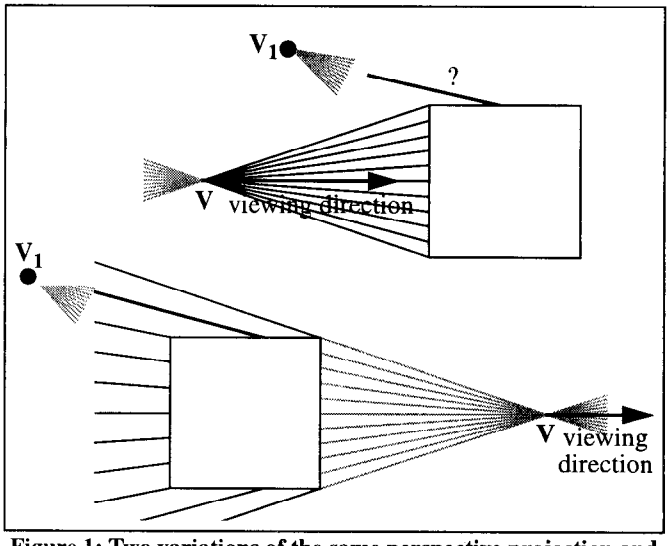

Figure 1: Two variations of the same perspective projection and their relevance for a new viewpoint V_1.

Nonetheless, perspective projections are flexible enough to capture more of an object than just a single side. Consider figure 1 which shows two perspective projections of a cube. The top projection samples only one side of the cube. This is the traditional use of perspective projection.
The same perspective projection can also be used to sample three (actually five in 3d) of the cube's faces at once. In this case, the rays diverging from the eye are used to project the object instead of the rays converging to the eye. The differ-

ence between the two projections is twofold:

- on the top the viewing direction is towards the object, on the bottom it is away from the object.
- on the top clipping restricts the depth of drawn points to an interval in front of the viewpoint, on the bottom this interval is behind the viewpoint.

Using this type of perspective projection a much larger portion of the object's surface is sampled than with a more traditional perspective projection. As a result, chances are lower to expose previously invisible portions of the object's surface when warping only one image for a new viewpoint.

When an image generated as on the top of figure 1 is warped to generate the view for point $\mathbf{V_1}$, the top of the cube will be missing. If the image generated as on the bottom of figure 1 is used instead, this surface was sampled and thus will appear in the warped image as well. When one thinks of every pixel in an image as a ray sampling the depicted object, images generated using traditional perspective projections contain no rays which appear in a perspective image of nearby viewpoints (except if moving exactly in the direction of view). In contrast, the bold ray on the bottom of figure 1 will also appear in the image for viewpoint $\mathbf{V_1}$.

Figure 2 shows a perspective projection of the teapot generated in this way. Note how this view includes both the top and the bottom in a single image. As a result, fewer images are necessary in an image-based object representation to cover the surface of the object.

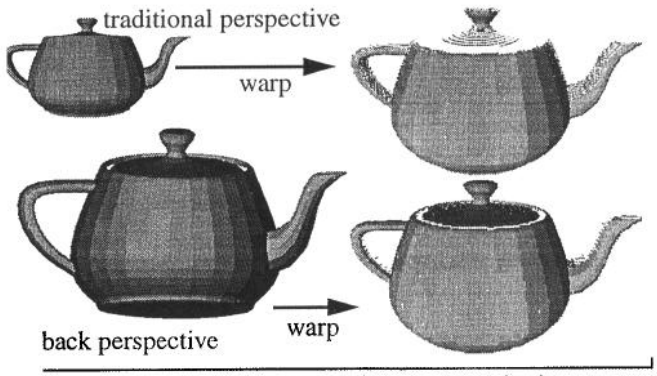

Figure 2: Top: warp of a traditional perspective image Bottom: warping a back perspective

The difference is obvious when warping the image for a new viewpoint (a view from a higher vantage point in the case given on the top of figure 2): the top is sufficiently represented and also the sampling on the handle and the spout is better.

It is apparent that the distortion in the bottom left image must be corrected during the warp towards the final image. In the approach presented here this is done with layered impostors.

3.2 Layered Impostors

Layered impostors [12] are a technique to warp a depth augmented image for a new viewpoint with hardware support on current graphics accelerators. They are based on the observation that the motion of each pixel due to changes of the point of view (the optical flow) can be approximated by rendering a number of texture-mapped polygons. The pixels in the image are grouped based on their depth and every group of pixels is drawn as a texture-mapped polygon at the corresponding distance from the new viewpoint. Depending on the number of layers used the image is warped with an accuracy increasing with the number of layers. The individual layers together approximate the volume of the viewing frustum which was used in the perspective projection. Figure 3 shows the teapot as an example of an object represented as an layered impostor. Note that again the apex of the viewing frustum was behind the teapot.

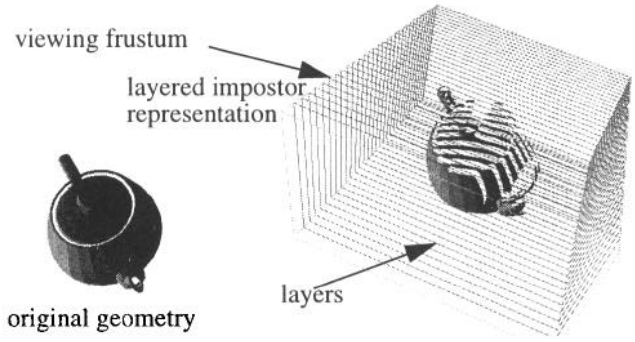

Figure 3: Layered impostor representation of the teapot

Each layered impostor consists of a stack of texture-mapped quadrilaterals and a texture with RGBz components per pixel. These components are stored as RGBα textures and the correct layer content is selected from the texture with the alpha-function (a function similar to the depth test which allows to selectively draw or reject pixels based on their alpha value). Blending is disabled for drawing layered impostors so that each pixel drawn is completely opaque. When drawing the quadrilaterals the proper texels are selected by testing for a certain depth (or α) value in the texture. The quadrilaterals are conveniently specified in screenspace and transformed back into world coordinates with the inverse of the perspective projection matrix which was used to generate the texture (the inverse of the perspective projection matrix mapping the viewing frustum in world coordinates to a cube in screen coordinates).

3.3 Avoiding Tears with Overlapping Depth Intervals

As can be seen on the right of figure 3 tears will eventually appear between the texels on each layer in the layered impostor under motion of the viewpoint. These tears can be postponed by not assigning disjoint depth intervals to each layer but overlapping ones. Consequently, the partial object images on the layers overlap as well and tears only appear under viewing conditions very different from the ones for which the image was generated. Figure 4 shows the difference and indicates how depth intervals overlap on the layers.

3.4 Generating the images in the representation

It appears to be an open research topic how to ensure a complete object representation by a number of images of an object. Previous approaches add all available images of real-world objects [7], distribute images evenly over the sphere of all possible viewing directions [3] or add all sufficiently different images [10]. For the approach presented in this paper fully automatic representation generation was a concern and so the approach of distributing images evenly over the sphere was chosen. In order to ensure such an even distribution, the sides of a platonic solid are considered and a view is either associated with each face normal, with each direction from a vertex to the centre or the solid or with all these directions. Figure 5 shows an example image-based object representation for a teapot (all images are square to facilitate texture definition in the graphics library used). Thirty-two images have been generated in total, of which only one half is shown due

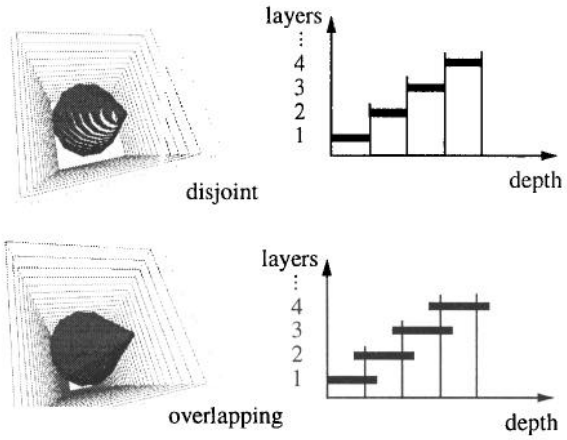

Figure 4: Layered impostor for a cone: depth intervals assigned to layers overlap to avoid tears between layers

to the symmetry of the teapot. On the bottom, a final image derived from this representation is shown together with the dodecahedron used to determine the viewing directions and its face normals. This image shows both how layered impostors warp an image and also produce approximate depth values at the same time. Traditional polygonal rendering (or line rendering in this example) can be mixed with images warped by layered impostors.

When distributing the images evenly over the sphere final images can be generated for arbitrary viewpoints and viewing directions. In the context of flight or vehicle simulators it might be sufficient to restrict oneself to a hemisphere of viewing directions or even to images showing the object along a horizontal viewing direction. In these cases only half or even fewer images need to be stored.

Real World Objects. For an image-based representation of real-world objects the special perspective projections must be derived once from the available images in a preprocessing reprojection step. Several images can be considered for one projection to avoid holes in the images of the representation.

Generating the final image. As was demonstrated in section 3.1 and shown in figure 2 it is sufficient to warp one image out of the ones available in the representation to obtain acceptable images of the object. The image chosen is the one which was generated with a viewing direction closest to the current viewing direction. Warping only one image has a performance advantage over warping several images. In cases where small holes in the final image are less tolerable (such as the one behind the handle of the top on the bottom of figure 5), several images can be warped, so that the holes are filled in from their samples (see below).

Avoiding excessive fillrate requirements. Layered impostors are drawn as a series of large textured polygons. They typically cover many pixels on screen and each pixel is potentially visited as often as there are layers in the layered impostor. In order to reduce the fillrate requirements of layered impostors the images in the representation can be examined to obtain a bounding rectangle for each layer which contains the object. Only these rectangles are drawn instead of the whole slices through the frustum. As a result empty layers can be left out and the area of the other layers is drastically reduced. Figure 6 left shows an example for another object where on each layer the rectangle bounding the object is shown.

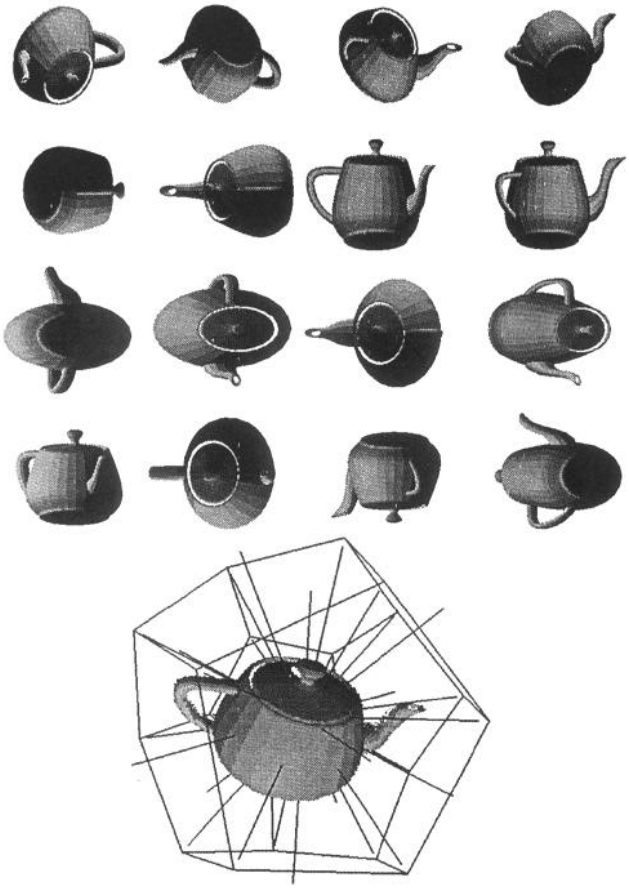

Figure 5: Views of the teapot generated for 32 viewing directions on a dodecahedron (in the directions from the vertices to the center and along face normals). Only half of the views are shown as the teapot is symmetric. Note how some views contain both bottom and top.

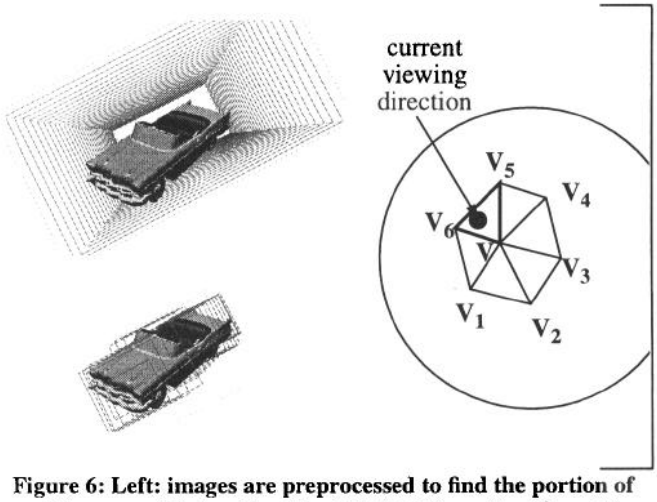

Figure 6: Left: images are preprocessed to find the portion of each layer containing the object. Right: Views V_i adjacent to view V on the sphere of possible viewing directions

Avoiding holes. When warping more than one image it is not necessary to draw all the layers in the layered impostors of similar viewing directions. It is sufficient to draw only those layers which will actually contain the texels to close the holes in the final image. These layers can be determined in advance

by rendering the layered impostor of view **V** once for each view $\mathbf{V_i}$ adjacent on the sphere of possible viewing directions (see figure 6 right) together with the layered impostor for view $\mathbf{V_i}$. Layered impostor $\mathbf{V_i}$ is rendered with a unique identifier instead of the texture's RGB colour. If holes appear in the warped image, layers of layered impostor $\mathbf{V_i}$ will be visible though them. These layers with identifiers appearing in the obtained image must be rendered in conjunction with the layered impostor for view **V**.
Consider the sphere of possible viewing directions given on the right of figure 6. For rendering a final image the triangle containing the current viewing direction is determined (say the bold one). The layered impostor for the viewing direction closest to the current viewing direction is rendered completely ($\mathbf{V_6}$). For the two other images in the triangle ($\mathbf{V_5}$ and **V**) only those layers are drawn which were found to fill in holes during preprocessing.

4 Implementation and Results

The presented image-based object representation has been implemented on top of the OpenGL graphics library and consequently runs on a variety of graphics platforms. Figure 7 shows an image of a scene rendered with an image-based representation for each of the objects (each containing from 5000 to 20000 polygons).

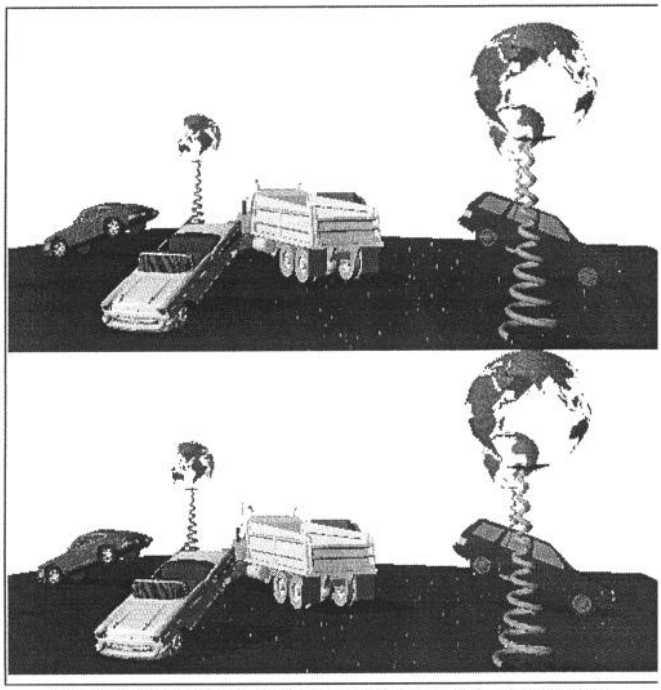

Figure 7: Top: image generated from original geometry Bottom: image generated using layered impostors for all objects except for the floor

As the image warping is done with layered impostors which in turn require textured polygons to be drawn, native hardware support for texture mapping is required for real-time performance. While the object representation could be based on any image-warping algorithm its memory requirements are independent of the warping algorithm used. These memory requirements are due to image storage and can be calculated as 4 * img_res * img_res * num_img. The factor 4 accounts for the RGBz components of the images. They are summarized in Table 2 for several numbers of images and image resolutions used.

Memory requirements for images in MB	image size 64x64	image size 128x128	image size 256x256
12 images	0.197	0.786	3.146
20 images	0.328	1.311	5.243
32 images	0.524	2.097	8.389

Table 2: Memory for the images in the representation

vertices in polygonal model of equal size	image size 64x64	image size 128x128	image size 256x256
12 images	5461	21845	87381
20 images	9102	36409	145636
32 images	14564	58254	233017

Table 3: Vertex count for polygonal model of equal size

With layered impostors as the image-warping methods acceptable results are obtained for twenty images or more. Otherwise the switching from one warped image to another is distractingly noticeable. With a different warping method fewer images might be sufficient.
This warping accuracy is achieved with 64 layers in the layered impostors. The implementation on top of OpenGL allows only powers of two for the number of layers. In order to store the bounding rectangles containing the object's image in each layer additional memory is required which can be calculated as 4 * num_img * layers (4 for four bytes storing the min/max extent in the image in four one-byte integers). Compared to the storage for the images this memory is quite negligible.
In order to decide when such an object representation is worth the cost one should compare these numbers to the storage requirements of traditional polygonal models. In OpenGL each vertex specification can include the coordinates (3 floats or 12 bytes), its normal (3 floats or 12 bytes), a vertex colour (4 bytes) and optional texture coordinates (2 floats or 8 bytes) summing to 24 bytes per vertex (or 36 bytes with texturing). The size of the required texture maps will vary widely, depending on whether the texture is tiled or not and mipmapped or not. Without counting the texture maps, Table 3 summarizes, for how many vertices (including texture coordinates) in a polygonal model, the image-based representation is more efficient. Of course more elaborate storage schemes for the images such as a delta tree [4] or image compression methods [1] would change these figures in favour of the image-based representation by a factor of ten. Unfortunately, such techniques are not currently available in hardware.
Figure 8 shows the drawing performance for layered impostors on a low-end hardware platform with texture-mapping support available in OpenGL. The SGI O2 graphics is capable of processing approximately 600000 vertices per second (an equivalent of 200000 independent triangles or 600000 triangles in long strips) and filling approximately 46 million pixels of textured triangles per second. The timings in the diagram were obtained by drawing a viewport-filling layered impostor on a viewport of sizes varying between 10 and 300 pixels. Drawing times have been measured using 64, 128 and 256 layers.
These diagrams can be used to determine, at what object complexity it is more efficient to draw an layered impostor representation instead of the geometry model. Usually, for geometry models the vertex performance of the graphics hardware determines the drawing speed (as the triangles are small enough so that vertex processing dominates the pixel-fill times). The horizontal dashed lines in the diagram indicate how long it takes to draw a model with the indicated number of independent triangles.

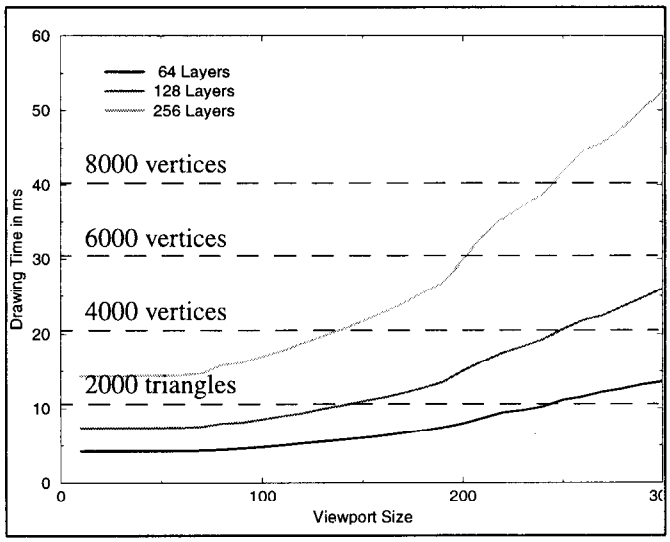

Figure 8: Drawing performance on O2 graphics hardware

As long as the object depicted on the layered impostor is more complex than the given vertex count and the graph remains below the dashed line for a given on-screen size of the layered impostor, it is more efficient to draw the layered impostor. For example, on O2 graphics and layered impostors with 64 layers, it is more efficient to draw the layered impostor representation for models exceeding 2000 triangles as long as the on-screen projection of the layered impostor covers less than 220 by 220 pixels.

Note how there is a horizontal section on the left of the graphs which is a result of the pipelined architecture of this graphics architecture. Vertex and pixel processing occur in parallel in a pipelined fashion. As long as the pixel processing time is smaller than the vertex processing time, pixel processing does not influence the graphics performance. In other words, primitive processing occurs with a speed determined by the slower of the two pipeline stages.

5 Conclusions and Future Work

This paper presented a new image-based object representation which (as other image-based representations) features independence of object complexity both in memory consumption and rendering time. It is based on an image-warping method which is supported in hardware on current graphics accelerators. The representation method differs from previously presented ones by the projection method used to obtain the images in the representation and by the image warping method applied. The combination of the two makes the method efficient both in memory consumption (as the number of images needed is minimized) and in drawing times (as graphics hardware can efficiently be employed).

This object representation is particularly suited to shift work in the graphics pipeline from the vertex processing stage to the pixel filling stage. Especially for complex distant objects this representation scheme can serve as a level of detail representation which is generated fully automatically.

In particular, the method has been optimized so that it is mostly sufficient to warp one image out of the representation to obtain new views of the object. In addition, a method has been presented how to efficiently fill the small holes caused by visibility changes by drawing just a few more textured polygons (layers in layered impostors of adjacent views). Fill-rate requirements have been reduced by finding the bounding rectangle which contains the opaque portion of the texture for each layer in the layered impostor.

A number of quality vs. storage requirements and drawing time trade-offs are possible: image resolution, number of images in the representation and warping accuracy can all be adapted to the application's needs.

Novel hardware platforms might support even more efficient implementations of the method. Rendering from compressed textures should cut down memory requirements by a factor of 10:1 to 30:1 [1][17]. Intelligent pixel-fill hardware could rapidly skip over transparent portions of the textures. Image-based rendering architectures [14] might soon offer better warping methods allowing a further reduction of the number of images used.

Future research should address the question how to ensure that an object has been completely represented by a number of depth-augmented images either by efficient implementations of multi-layer depth images or by ensuring that every portion of the object's surface is sufficiently sampled in at least one of the images.

References

[1] Beers, Andrew C., Maneesh Agrawala, Navin Chaddha, *"Rendering from Compressed Textures"*, SIGGRAPH '96, pp 373-378.

[2] Chen, Shenchang Eric and Lance Williams, *"View Interpolation for Image Synthesis"*, SIGGRAPH '93, pp 279-288.

[3] Chen, Shenchang Eric, *"Quicktime VR - An Image-Based Approach to Virtual Environment Navigation"*, SIGGRAPH '95, pp 29-38.

[4] Dally, William J., Leonard McMillan, Gary Bishop, and Henry Fuchs, *"The Delta Tree: An Object-Centred Approach to Image-Based Rendering"*, MIT AI Lab Technical Memo 1604, May 1996.

[5] Darsa, Lucia, Bruno Costa, Amitabh Varshney, *"Walkthroughs of Complex Environments using Image-based Simplification"*, Symposium on 3D Interactive Graphics '97, April 27 - 30, 1997, Providence, RI, pp 25 - 34.

[6] Debevec, Paul E., Camillo J. Taylor and Jitendra Malik, *"Modelling and Rendering Architecture from Photographs"*, SIGGRAPH '96, August 1996.

[7] Gortler, Steven J., Radek Grzeszczuk, Richard Szeliski and Michael F. Cohen, *"The Lumigraph"*, SIGGRAPH '96, pp 43-54.

[8] Gortler, Steven J., Li-wei He and Michael F. Cohen, *"Rendering Layered Depth Images"*, Microsoft Technical Report MSTR-TR-97-09, March 1997.

[9] Levoy, Marc and Pat Hanrahan, *"Light Field Rendering"*, SIGGRAPH '96, pp 31-42.

[10] Maciel, Paulo W. and Peter Shirley, *"Visual Navigation of Large Environments Using Textured Clusters"*, Symposium on Interactive 3D Graphics (April 1995) pp 95-102.

[11] Max, Nelson, *"Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers"*, Proceedings of the 7th Eurographics Workshop on Rendering '96, Porto, Portugal, pp165-174.

[12] Schaufler, Gernot, *"Per-Object Image Warping with Layered Impostors"*, Proceedings of the 9th Eurographics Workshop on Rendering '98, Vienna, Austria, pp 145-156.

[13] Sillion, François X., George Drettakis and Benoit Bodelet, *"Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery"*, Proceedings of Eurographics'97, September 4-8, 1997, pp 207-218.

[14] Mark, William R., Gary Bishop, *"Memory Access Patterns of Occlusion-Compatible 3D Image Warping"*, Proceedings of the 1997 SIGGRAPH / Eurographics Workshop on Graphics Hardware (Los Angeles, California), August 3-4 1997, pp. 35-44.

[15] McMillan Leonard and Gary Bishop, *"Plenoptic Modelling: An Image-Based Rendering System"*, SIGGRAPH '95, pp 39-46.

[16] Pulli, Kari, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro, and Werner Stuetzle, *"View-based Rendering: Visualizing Real Objects from Scanned Range and Colour Data"*, Proceedings of 8th Eurographics Workshop on Rendering, St. Etienne, France, June 1997, pp 23-34.

[17] Torborg, Jay and Jim Kajiya, *"Talisman: Commodity Real-Time 3D Graphics for the PC"*, SIGGRAPH '96, pp 353-363.