Image-based rendering for real-time applications

Matthias Bauer*

Media Technology and Design
University of Applied Sciences Hagenberg
Austria

Abstract

Due to new techniques and new hardware capabilities image-based rendering (IBR) is not limited to offline rendering anymore. Today the use of pictures for complex 3d-models promises photo-realistic images even for real-time applications.

This paper covers the various techniques which are capable of creating new perspectives in real-time. It also introduces a framework which enables the convenient inclusion of image-based objects into 3d-applications.

Keywords: image-based rendering, real-time, 3d

1 Introduction

In image-based rendering[1] photographs or artificial images are used as source for three-dimensional objects or complete scenes. This technique has certain advantages over a traditional graphics pipeline because it is often easier to photograph an object than to model it using 3d-creation software. So-called photo realism has always been an aim of computer graphics and IBR promises this per definition.

As of now many different techniques have been developed that are based on images as source material. Most of them are used in the domain of offline renderings, for which the duration of the needed calculations for one frame is not as relevant. Because of the rapid development in the field of graphics hardware, those techniques become more and more interesting for real-time applications for which 20 or more frames per second are necessary.

Traditional 3d-applications such as many video games could greatly benefit from IBR through a higher degree of realism. But IBR also offers completely new possibilities. For example in the field of communications 3d-telepresence becomes possible. Through the combination with augmented reality conversational partners could see each other face to face in a completely realistic way without sharing the same location [17].

The advantage of IBR boils down to more realistic results with less work. Less work because of the modeling which can be mostly omitted and because of the lack of complex models which would need more computing power. Images also already include materials and lightning conditions, which reduces the necessary manual work and the render complexity even further.

On the other hand the predefined materials and lights present limitations for IBR because it is far more difficult to change them afterward. Debevec et al. [6] present a method to generate pictures of faces from arbitrary points of view and under different lightning conditions, but the enormous amounts of data and the high computational costs make it impractical for real-time applications on today's hardware.

A further problem is the animation of IBR-objects. While some systems allow the playback of "3d-videos" in real-time, in which the observer can move freely around the recorded object, they still cannot offer the possibilities of traditional computer graphics. Interactive motions like they are essential for 3d-avatars or game characters are not possible with today's IBR-techniques.

The use of IBR in 3d-applications is not only dependent on the technical constraints but also on the simple access to the field. Just like the shader language Cg has been developed with the thought in mind to make the new possibilities of graphics cards available to more programmers [14], there need to be easy interfaces to integrate IBR into applications.

So this paper will discuss the various IBR-techniques and examine their real-time capabilities. Then we will propose an IBR-framework and present a simple prototype that uses it to render three-dimensional objects. The achieved quality and performance will show the potential for IBR in real-time applications.

2 Real-time IBR techniques

This section introduces various image-based techniques. It is not a complete overview of IBR but focuses on methods with some sort of real-time capability.

*matthias.bauer@fh-hagenberg.at

[1] IBR is a subgroup of image-based modeling and rendering. However, image-based modeling (IBM) creates 3d-objects from pictures which are then usually rendered in a traditional way. Although there are techniques that combine IBM and IBR, this paper tries to focus solely on IBR in which pictures can directly be used as source for new renderings.

2.1 Sprites and billboards

Two-dimensional images gained importance in the field of computer graphics during the eighties. The use of texture mapping added more realism to interactive applications without the need for additional geometry. However, usual textures just give a certain appearance to a surface but they cannot substitute 3d-objects by themselves.

A simple method to use images instead of geometry is billboards. Billboards are basic quads which are always kept aligned with the user's view. If an image is mapped on a billboard it gives the impression of an object which looks the same from all perspectives. The so-called sprites were often used to display characters in video games before the hardware could handle more complex models that looked realistic enough. Another application is trees and other plants which would cost much computational power to render as geometries but are actually only circumstantial for the program.

Today billboards are widely used for all sorts of effects like fog, smoke, flames, lens flares, and various particle systems. Sprites which are not aligned to any view axis can also be used to represent complex objects, especially plants because of their more chaotic structure (see figure 1).

Figure 1: The complex structure of a bush can be approximated by interweaving several sprites. Image from the video game *Medal of Honor*.

2.2 Photo mosaics

A well-known commercial application of IBR is Apple's QuicktimeVR[2] [4]. This technology uses a 360° panorama mapped on a cylinder. The user is positioned at the center and can rotate the view around the vertical axis. Through warping a perspectively correct image is generated. Newer versions of QuicktimeVR also support cube maps which allow the viewer to look into all directions. In 3d-applications cube or environment maps are commonly used to display backgrounds or simulate reflections [3, 10].

Lippman already presented an adoption of this technique in 1980 [13]. His application – called *Movie Map* – was made possible by high data capacities of video discs. The system allowed a stroll through Aspen but as in QuicktimeVR one could just stop at predefined positions. Current implementations use video cameras to record panoramas of an environment. Thus it is possible to stop at any position along a predefined path and have a 360° view available [25].

All these applications of photo mosaics are suitable for real-time rendering. Panoramas are often viewed via plugins for web browsers. Even Java versions[3] exist and they pose no problem for the computing power of today's PCs.

2.3 3d-warping and view interpolation

3d-warping of an image is possible if the original camera parameters and depth information are known. Through rotation and translation every pixel can be mapped onto a new perspective. This forward warping approach causes a few problems though, because more than one pixel can map to the same point in the destination image and gaps can occur because of too few or missing information in the source image. While the first problem can be solved efficiently with a warping order depending on the centers of projection [15], the gaps can only be filled by a heuristic approach.

Inverse warping [15] maps pixels in the destination image to the source image. This is an iterative process though because the depth value is unknown. So a ray is casted from the center of projection through the desired pixel. This ray is then projected onto the source image and the resulting line has to be checked iteratively for a corresponding depth value (see figure 2). Although this approach is computationally more expensive than forward warping, there have been several optimizations [16] and interactive frame rates are possible [26]. It also offers certain advantages in terms of depth mapping and gap filling.

The warping approach can be improved by using more than one reference image. Thus gaps can be filled from additional sources and if more than one reference pixel is visible in the desired view the color values can be interpolated to achieve better results. This technique reminds us of traditional morphing [2]. Morphing relies on corresponding features in reference images which was introduced to 3d with warp scripts [5]. However, this only allowed images to lie on the same plane. To overcome this problem view morphing [22] reduces the three-dimensional view interpolation to a two-dimensional process by adding warping steps before and after the actual morph. Because of this there is also no depth information but only camera parameters needed.

[2] `http://www.apple.com/quicktime/products/qt/overview/qtvr.html`

[3] e.g. `http://infomedia.ipix.com/`

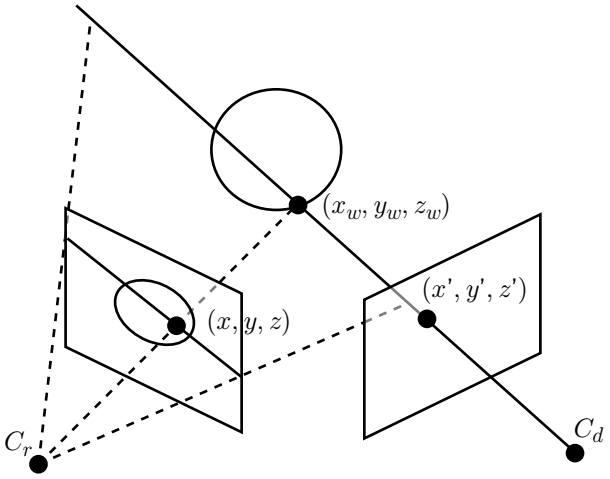

Figure 2: Inverse 3d-warping maps the point $(x', y') - z'$ is unknown – on the desired image to a point on the reference image by projecting a ray onto the reference image and iteratively checking depth values along the projected line.

2.4 Impostors and nailboards

So-called impostors try to utilize the similarities between frames. Whole scenes or parts of them are normally rendered into a buffer which then can be used as a texture on a billboard. This can be displayed instead of the actual geometry as long as only the line of sight changes. If the position of the viewer changes a new texture has to be rendered or another precalculated impostor is used. Usually the impostor does not have to be exchanged immediately when the viewer moves but only when a certain threshold is reached, so that textures don't have to be enlarged too much or the billboard's planarity does not become apparent [18].

Complete rendering systems based on impostors have been developed. They hierarchically subdivide a scene into cuboids and render impostors for each one. It is especially crucial to determine when an impostor has to be rerendered. The hierarchical partitioning further accelerates the rendering process. Those image caching systems can be two to eight times faster than traditional systems. However, these are results from offline renderings [21, 24].

Since impostors are directly derived from geometry it is simple to save an additional depth value for each pixel. The result is a nailboard [19] which can be used longer because of 3d-warping. The extra depth channel also solves visibility problems which occur when impostors cut each other. Enhanced versions like layered impostors [20] or layered depth images [23] use layers of nailboards for certain depth ranges. The advantage of this is that often just specific layers have to be rerendered. [8] combines dynamic and precalculated impostors into a framework called multi-mesh impostors. It achieves interactive frame rates and ten times the speed of traditional rendering methods.

2.5 Lumigraph and light fields

Since more reference images improve the quality of rendered views, some methods try to capture objects with an extensive amount of images. Adelson and Bergen propose a five-dimensional plenoptic function which describes all incident rays of light at a single point. The parameters are the position (x, y, z) and two angles that specify the light direction [1]. The Lumigraph [9] and light field renderings [12] advance this concept. Both reduce the function to four dimensions which is possible through a reduction to the convex hull of an object and the assumption that light remains constant along a ray.

Basically these methods try to have an image for all possible perspectives and interpolate in between if necessary. This results in vast amounts of data. Similarities between the reference images allows high compression rates though. For example [12] reports ratios of up to 118:1. This makes these techniques suitable for real-time applications too. There are no complex calculations involved and it is mostly a matter of look-up speed and memory size.

3 IBR-framework

The idea behind the presented framework is to have an *IBRObject* which can be added to a scene graph and thus enabling the inclusion of objects which rely solely on images instead of geometry. To display a complex object from all sides a single image is not sufficient. So an *IBRObject* has to hold an array of images. We define an *IBRImage* to hold additional information such as camera parameters which are necessary for most IBR-techniques. The actual algorithm for rendering should be exchangeable and of no concern for the programmer who just uses the framework as a library to include IBR-models (see figure 3).

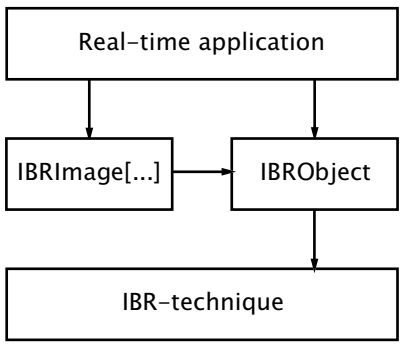

Figure 3: The basic design of the IBR-framework.

Using this framework it would be very easy to integrate an image-based object but the acquisition of suitable images can still be difficult without special camera equipment. The easiest way to apply IBR to real-time ap-

plications is to use computer-created imagery. Using 3d-modeling software an artist creates a high-polygon model of the desired object. Then with a virtual camera which has known parameters the reference images are created. This has also the advantage that it fits into the traditional workflow.

3.1 The prototype

To demonstrate the framework we implemented a prototype which uses projective mapping and blending as a simple IBR-technique (see figure 4). Projective mapping means that the reference image is warped according to its original viewing parameters. Jakulin's similar approach for displaying plants [11] uses a quad for every reference image. Thus the regular rendering pipeline takes care of the warping. We use a single billboard instead and render the current view onto it. To achieve this in one render pass we make use of a custom fragment shader that warps every fragment onto each reference image and blends the resulting color values.

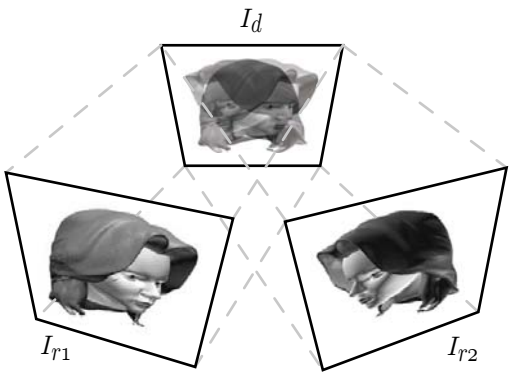

Figure 4: The two closest reference images I_{r1} and I_{r2} are warped according to their viewing parameters, mapped onto a billboard I_d, and blended to create the desired view.

We chose this approach because it would be easier to exchange the IBR-technique if their is only one final billboard to render to. Methods that use multiple quads or other techniques can always render to this billboard's texture in a final step. However, it is necessary to avoid unwanted clipping that can occur this way (see figure 5).

For the actual blending the color values of the warped reference images are weighed according to the angles between the reference viewing directions and the current viewing direction. For example for the angles α and β we can get the weight factors f_α and f_β with the equations:

$$f_\alpha = \frac{\alpha}{\alpha+\beta}, f_\beta = \frac{\beta}{\alpha+\beta}.$$

The weighted color values are then added up to get the final color.

The prototype was written in C++ and the shaders in Cg. As low level graphics engine we used OpenGL. Those

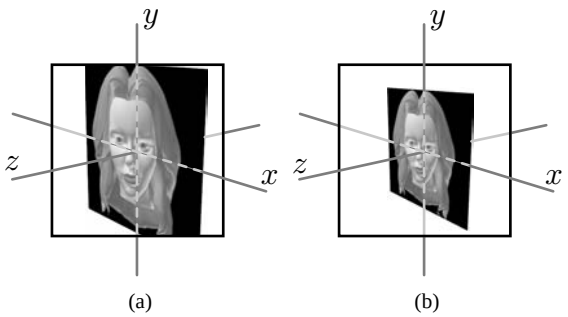

Figure 5: (a) shows unwanted clipping that can occur when warping a reference image onto the billboard. The prototype allows the programmer to define a scale factor. In (b) a factor of 0.7 was used so that the warped image fits completely.

technologies were chosen to guarantee a high degree of platform independence.

Furthermore *IBRViewer* served as a sample application. It simply displays one or several *IBRObjects* and lets the user move around and view the objects from every angle.

4 Results

This section describes the various tests cycles completed with *IBRViewer*. The two main criteria are the quality of the produced renderings and the speed when rendering multiple objects. The used system was a Pentium 4 (2.53 GHz) with 1 GB of memory. The graphics card was an NVIDIA GeForce FX 5900XT with 128 MB of memory. It supports vector and fragment shader version 2.0 which was important because of the Cg programs. All renderings had a resolution of 400x400 pixels. The reference images had a size of 256x256 pixels and were created with 3ds max.

4.1 Quality

The most important factor to achieve a good quality is the amount of reference images. Figure 6 shows the test cycle with the model of a human head. The more uniform the distance between the surface and the center of the model is, the better it is suited for this technique because highly convex areas change the most from one reference image to the next. The head partially fulfills this but definite lines in the face cause problems. For example the hairline stands out when it is blended right on top of the eye. This ghosting effect is much less visible with more uniform surfaces like the back of the head. So to improve the quality it would be possible to adjust the positions of the reference cameras and use more images for the front than for the back.

The test cycle used 16 to 64 reference images which

Figure 6: The picture on the left was created using 16 reference images, the one on the right using 64.

were evenly distributed on the same plane. So it is possible to move around the object as long as we stay on this plane. With 64 images the blending already looks very much like a motion blur when moving which could be a quite desirable effect. Generally the motion is important for a more realistic result. If we circle the object at high speed 16 reference images turned out to be sufficient. It would be possible to make the amount of images dependent on the speed of the viewer and just render the closest reference view if the viewer is not moving at all.

The model for the second test cycle was a potted plant (figure 7). This model is interesting because the pot is perfectly symmetric and the plant has a very complex structure. The symmetry and the uniform texture of the pot allow good results even with very few reference images. The complex structure of the plant makes it difficult to grasp it completely. Thus single leaves and branches that radically change their positions from one image to the next hardly stand out. The viewer can just follow the blending at very distinct branches.

Figure 7: The picture on the left was created using 16 reference images, the one on the right using 64.

So for this example fewer reference images are necessary than for the head. The number of images can be reduced further at higher distances. This could be incorporated in a level-of-detail algorithm which discards reference images when the viewer moves away from the object.

4.2 Speed

With a single IBR-object and 64 reference images the prototype achieved 77 frames per second (fps). This seemed to be the highest measurable value because reducing the number of reference images had no effect anymore. With 100 objects displayed simultaneously the frame rate dropped to 11 fps. However the bottleneck was not the GPU and the actual graphics processing but the sorting of the images to determine the closest reference views. Without it there were still 38 fps possible. Thus it would be desirable to organize the images in some kind of “view map” [7] which allows a simple look-up of the closest image. So the current implementation is very much dependent on the total number of reference images in a scene. A test with 100 objects and only 16 images also shows this because it still allowed 29 fps.

For the tests the same object was duplicated. That means that all instances of it access the same textures. So a more complex scene with many different objects could lead to a shortage of graphics memory. Then either the total number has to be reduced using the described methods or techniques for compression and dynamic loading of image data have to be applied (cf. [9, 12]).

5 Conclusions and future work

The current implementation is actually just a very simple prototype. Still it is apparent that with today’s hardware it is possible to integrate IBR-objects into real-time applications. However, this is just reasonable if it provides advantages in the development process and allows an efficient workflow.

Section 4 already stated several approaches to achieve better results. For better quality more reference images are necessary but this means a higher computational cost even if the sorting is optimized. It also raises the need for memory and the acquisition of the source material can be very laborious. Even with computer generated imagery it can be a time-consuming process. Special plug-ins for current tools could simplify this but with the use of photographs the work increases drasticly.

So to achieve better quality without raising the number of reference images, other techniques to combine the images are necessary. A promising approach would be view morphing, which would not need any additional information beside the camera parameters. Inverse 3d-warping would be easy to implement as well with newer hardware and shader version 3.0, which allows iterative processes in fragment programs. However, this technique would need an additional depth channel. Certain extensions could also be added – especially through the use of shaders. Examples would be shadows or lightning with the use of normal maps.

The use of many images provides an intricate inclusion process for the programmer. So all images could be com-

bined in a single file with its own IBR-format which holds the picture data and also camera parameters and other meta information – for example to enable LOD-algorithms. An IBR-format would change the framework in a way that the programmer just has to handle the *IBRObject*. *IBRImage* would only be used internally to manage reference images.

The loading of images could be abstracted further because other sources are possible. For example video streams could allow animated objects which are essential for 3d-telepresence.

In general the use of IBR in real-time applications will be significantly dependent on the availability of those techniques for designers and programmers. Because only if they are accessible through tools, programming libraries, and standards, IBR can be a useful addition to computer graphics.

References

[1] Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J. A. Movshon, editors, *Computation Models of Visual Processing*, chapter 1, pages 3–20. MIT Press, Cambridge, MA, 1991.

[2] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Proceedings of SIGGRAPH*, pages 35–42, Chicago, Juli 1992.

[3] James F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of SIGGRAPH*, pages 192–198, San Jose, Juli 1977.

[4] Ercic Chen. Quicktime vr: an image-based approach to virtual environment navigation. In *Proceedings of SIGGRAPH*, pages 29–38, Los Angeles, September 1995.

[5] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of SIGGRAPH*, pages 279–288, Anaheim, August 1993.

[6] Paul Debevec, Tim Hawkinsy, Chris Tchouy, Haarm-Pieter Duikery, Westley Sarokiny, and Mark Sagarz. Acquiring the reflectance field of a human face. In *Proceedings of SIGGRAPH*, pages 145–156, New York, Juli 2000. ACM Press/Addison-Wesley.

[7] Paul Debevec, Yizhou Yu, and George Borshukov. Efficient viewdependent image-based rendering with projective texturemapping. In *Proceedings of Eurographics Workshop on Rendering*, pages 105–116, Wien, Juni 1998.

[8] Xavier Decoret, François Sillion, Gernot Schaufler, and Julie Dorsey. Multi-layered impostors for accelerated rendering. *Computer Graphics Forum*, 18(3), 1999.

[9] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH*, pages 43–54, New Orleans, August 1996.

[10] Ned Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11), November 1986.

[11] Aleks Jakulin. Interactive vegetation rendering with slicing and blending. In *Proceedings of Eurographics 2000 (Short Presentations)*, Interlaken, August 2000.

[12] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH*, pages 31–42, New Orleans, August 1996.

[13] Andrew Lippman. Movie-maps: An application of the optical videodisc to computer graphics. In *Proceedings of SIGGRAPH*, pages 32–42, Seattle, Juli 1980.

[14] William R. Mark, R. Steven Glanville, Kurt Akeley, and Mark J. Kilgard. Cg: a system for programming graphics hardware in a c-like language. *ACM Transactions on Graphics*, 22(3):896–907, 2003.

[15] Leonard McMillan Jr. *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis, University of North Carolina, Chapel Hill, 1997.

[16] Robert W. Mercato. Optimizing an inverse warper. Master's thesis, MIT, Department of Electrical Engineering and Computer Science, Cambridge, MA, Mai 1998.

[17] Simon Prince, Adrian David Cheok, Farzam Farbiz, Todd Williamson, Nik Johnson, Mark Billinghurst, and Hirokazu Kato. 3d live: Real time captured content for mixed reality. In *Proceedings of ISMAR*, pages 7–13, Darmstadt, September 2002.

[18] Gernot Schaufler. Dynamically generated impostors. In D. W. Fellner, editor, *GI Workshop "Modeling - Virtual Worlds - Distributed Graphics"*, pages 129–135. infix Verlag, November 1995.

[19] Gernot Schaufler. Nailboards: A rendering primitive for image caching in dynamic scenes. In *Proceedings of Eurographics Workshop on Rendering*, pages 151–162, St. Etienne, Juni 1997.

[20] Gernot Schaufler. Image-based object representation by layered impostors. In *Proceedings of ACM Symposium on Virtual Reality Software and Technology*, pages 99–104, Taipei, November 1998.

[21] Gernot Schaufler and Wolfgang Stürzlinger. A three dimensional image cache for virtual reality. In *Proceedings of Eurographics*, pages 227–236, Poitiers, August 1996.

[22] Steven M. Seitz and Charles R. Dyer. View morphing. In *Proceedings of SIGGRAPH*, pages 21–30, New Orleans, August 1996.

[23] Jonathan Shade, Steven Gortler, Li-wei Hey, and Richard Szeliskiz. Layered depth images. In *Proceedings of SIGGRAPH*, pages 231–242, Orlando, Juli 1998.

[24] Jonathan Shade, Dani Lischinski, David H. Salesin, Tony DeRose, and John Snydery. Hierarchical image caching for accelerated walkthroughs of complex environments. In *Proceedings of SIGGRAPH*, pages 75–82, New Orleans, August 1996.

[25] Matthew Uyttendaele, Antonio Criminisi, Sing Bing Kang, Simon Winder, Richard Hartley, and Richard Szeliski. High-quality image-based interactive exploration of real-world environments. *IEEE Computer Graphics and Applications*, 24(3), 2004.

[26] Xin Zheng and Enhua Wu. Efficient 3d image warping for composing novel views. In *Proceedings of Computer Graphics International*, pages 123–130, Hong Kong, Juli 2001. IEEE Computer Society.