# Multiple Eigenspaces for Hardware Accelerated Image Based Rendering

Markus Grabner [1], Horst Bischof [1],
Christopher Zach [2], Andrej Ferko [1]
[1] ICG, TU Graz, Austria, [2] VRVis Research Center, Graz, Austria
{grabner,bischof,zach,ferko}@icg.tu-graz.ac.at

*Abstract:*
*We present a novel hardware accelerated method for rendering images of a 3D scene. The method uses the recently proposed multiple eigenspace method to obtain an efficient representation of rotated objects that can be easily implemented on modern graphics cards using their fragment shader capabilities. Due to the restrictions of current graphics hardware (which are discussed in detail), the multiple eigenspaces method (originally proposed for the purpose of object recognition) has to be slightly modified. We use the Stanford Bunny model to evaluate our method. Our experiments demonstrate the excellent performance of hardware-based image reconstruction.*

## 1 Introduction

A variety of so-called *image-based rendering* (IBR [9]) techniques has been developed to solve the problem of photorealistic rendering. The basic idea is to replace complex geometry and illumination by an image of the surface (or, more recently, multiple images). These images can be precomputed using sophisticated rendering techniques with almost no restriction in computational complexity. Alternatively, images of an object can be taken directly in its natural environment (or under laboratory conditions for better control of illumination), thus bypassing the 3D reconstruction stage entirely.

If the user should be allowed to view the scene from any position and direction, the complete five-dimensional *plenoptic function* [9] needs to be sampled (and properly reconstructed for rendering). In the absence of obstructions, this function can be reduced to a four-dimensional function, called *Lumigraph* [6] or *light field* [8]. A much simpler approach is the *billboard* concept [3]. It uses a single two-dimensional image that is rotated (with one or two degrees of freedom) to be always oriented towards the viewpoint. Billboards are only suitable for objects that are almost rotationally symmetric (such as trees and similar objects). This technique is implemented in the wide-spread VRML language [11].

As an intermediate solution between the billboard concept and sophisticated approximations of the plenoptic function, we present a technique that captures the object's appearance with one degree of freedom (i.e., a rotation around the vertical axis). Thus a three-dimensional function $f(x, y, \varphi)$ is created, where $\varphi$ is the angle of rotation. To reduce the amount of data, compression is done in our method by *multiple eigenspaces decomposition* [7] of the different views of the object. The final image is composited by graphics hardware, such as recent boards by NVidia and ATI. The fragment shader capabilities of these boards are accessed by the language "Cg" [1]. This allows almost platform-independent code. Moreover, we expect our method to fit nicely and efficiently into a polygonal rendering environment.

This paper is organized as follows. Related methods are shortly reviewed in Section 2. Details on our method are given in Section 3, experimental results are presented in Section 4. The paper is concluded in Section 5.

## 2 Related work

### 2.1 Multiple eigenspaces

A correlated set of images can be compressed by using eigenspaces [10]. The obtained compression will be quite good if the whole set of images is highly correlated. However, if the object to be compressed has a different appearance from various view-points, the obtained compression is not optimal. To alleviate that problem, a novel self-organizing framework to construct multiple, low-dimensional eigenspaces from a set of training images has recently been proposed [7]. Grouping of images is systematically and robustly performed via *eigenspace-growing* in terms of low-dimensional eigenspaces. To further increase the robustness, the eigenspace-growing is initiated independently with many seeds (small groups of images). All these grown eigenspaces are treated as hypotheses that are subject to a selection procedure *eigenspace-selection*, based on the minimum description length principle (MDL), which selects the final resulting set of eigenspaces as an efficient representation of the training set, taking into account the number of images encompassed by the eigenspaces, their dimensions, and their corresponding residual errors.

In this framework each image $\mathbf{x}$ can then be represented as a linear combination of eigenimages forming an eigenspace: $\mathbf{x}_i = \sum_{j=1}^{m} \mathcal{I}_j^{(i)} \sum_{k=1}^{d_j} c_{jk}^{(i)} \mathbf{e}_{jk}$, $m$ is the number of all eigenspaces, $d_j$ is the dimension of the $j$-th eigenspace, $\mathcal{I}_j^{(i)}$ is a variable which is 1 for the $j$-th eigenspace which encodes the image $\mathbf{x}_i$ and 0 otherwise, and $c_{jk}$ and $\mathbf{e}_{jk}$ are the corresponding coefficients and eigenimages, respectively. All the required parameters are found completely automatically.

### 2.2 Image-based rendering

Plenoptic modeling [9] has been introduced as a general framework for image-based rendering. The authors state that the goal of any image-based rendering system is the generation of a continuous representation of the plenoptic function, given a set of discrete samples. An informed survey on IBR can be found in [4].

If we are only interested in a single object without obstructions, a four-dimensional representation of the light flow is sufficient, as it was independently presented by Gortler [6] and Levoy [8] at the same conference. Both authors state a huge amount of data required to represent even the reduced four-dimensional light distribution function. Although compression ratios of better than 1:100 have been reported [6, 8], the resulting data sets (several megabytes in size) are still prohibitive for remote visualization purposes over any network except high-speed LANs.

In the dynamic textures approach [5], small details on a (polygonal) surface are captured by principal component analysis. While the coarse geometry is rendered traditionally, surface details are applied by image based rendering. A hardware implementation of the algorithm using texture blending is proposed. The method requires up to 100 texture bases, in contrast to the rather small eigenspace dimension in our method.

## 3 Our Method

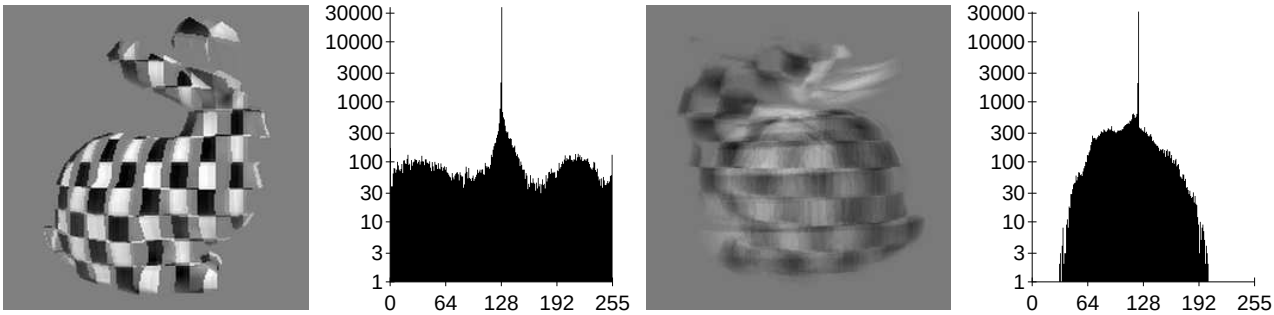
### 3.1 Eigenspaces construction

Input images are assigned to eigenspaces by performing the calculations with scaled down color images. Once the assignment is determined, the computation is done in full resolution.

In order to use multiple eigenspaces on graphics card we have to take into account the limited number of texture buffers available. In addition we require that all eigenspaces have a similar reconstruction error (i.e., in order to reduce annoying visualization artifacts when moving from one eigenspace to another). These requirements can be easily incorporated in the eigenspace growing process.

### 3.2 Hardware-based image reconstruction

Graphics adapters offering at least some limited programmability of their fragment creation stage have become commonplace during the past years. However, the flexibility largely differs between graphics cards from different vendors and even between different models of the same vendor. The “Cg” language was designed by NVidia to overcome these problems [1]. It




(a) eigenimage 3/1 (b) histogram of 1(a) (c) eigenimage 5/1 (d) histogram of 1(c)

**Figure 1: Eigenimages (256×256 pixels, green channel) scaled to the range of texture images and the corresponding histograms**

is a high-level language (similar to the popular "C" programming language) to encapsulate hardware details in a platform-specific compiler back-end. However, to achieve optimal results, the programmer should be aware of limitations of the target platform.

The following discussion refers to NVidia's GeForce4 since it fulfills the minimum requirements for image reconstruction from multiple eigenspaces. Eigenimages and the mean image have to be stored as texture images with an integer range from 0 to 255. Numeric operations in the fragment shaders are performed in fixed point registers of eight bits plus one sign bit. Mapping the input range from $[0; 255]$ to $[-1; +1]$ is supported in hardware without additional computation time. Similar considerations apply to the coefficients, which are passed as the RGBA components of a color register. The number of texture images that can be processed simultaneously is limited to four on the GeForce4.

Now let us examine how the image reconstruction equation

$$\mathbf{y} = \sum_{i=0}^{N} c_i \mathbf{x}_i \tag{1}$$

for a single eigenspace will be processed in hardware. Note that for uniformity we consider the mean image to be given by $\mathbf{x}_o$ with a constant coefficient $c_0 = 1$. The remaining coefficients $c_i$ for the eigenimages $\mathbf{x}_i$ are linearly interpolated between the values computed for the input orientations.

Equation 1 cannot be calculated exactly for two reasons. First, the number of texture images that can be processed in a single render pass is limited to four, therefore only the mean image and three eigenimages can be used (we define $M = \min(N, 3)$ in our case). It is straightforward to incorporate this limitation into the multiple eigenspaces framework by using a maximum eigenspace dimension in the growing step of the algorithm as indicated in Section 3.1. Second, due to the limited number of bits, calculations are performed with quantized values $\tilde{c}_i = c_i + q_i$

```
#include "structs.h"

float4 main(VertexOut In,
    uniform sampler2D img0,
    uniform sampler2D img1,
    uniform sampler2D img2,
    uniform sampler2D img3): COLOR
{
  return
    4 * (In.coeffs[0] - 0.5) * (tex2D(img0, In.texCoord0.xy) - 0.5) +
    4 * (In.coeffs[1] - 0.5) * (tex2D(img1, In.texCoord1.xy) - 0.5) +
    4 * (In.coeffs[2] - 0.5) * (tex2D(img2, In.texCoord2.xy) - 0.5) +
    2 * (In.coeffs[3] - 0.5) *  tex2D(img3, In.texCoord3.xy);
}
```

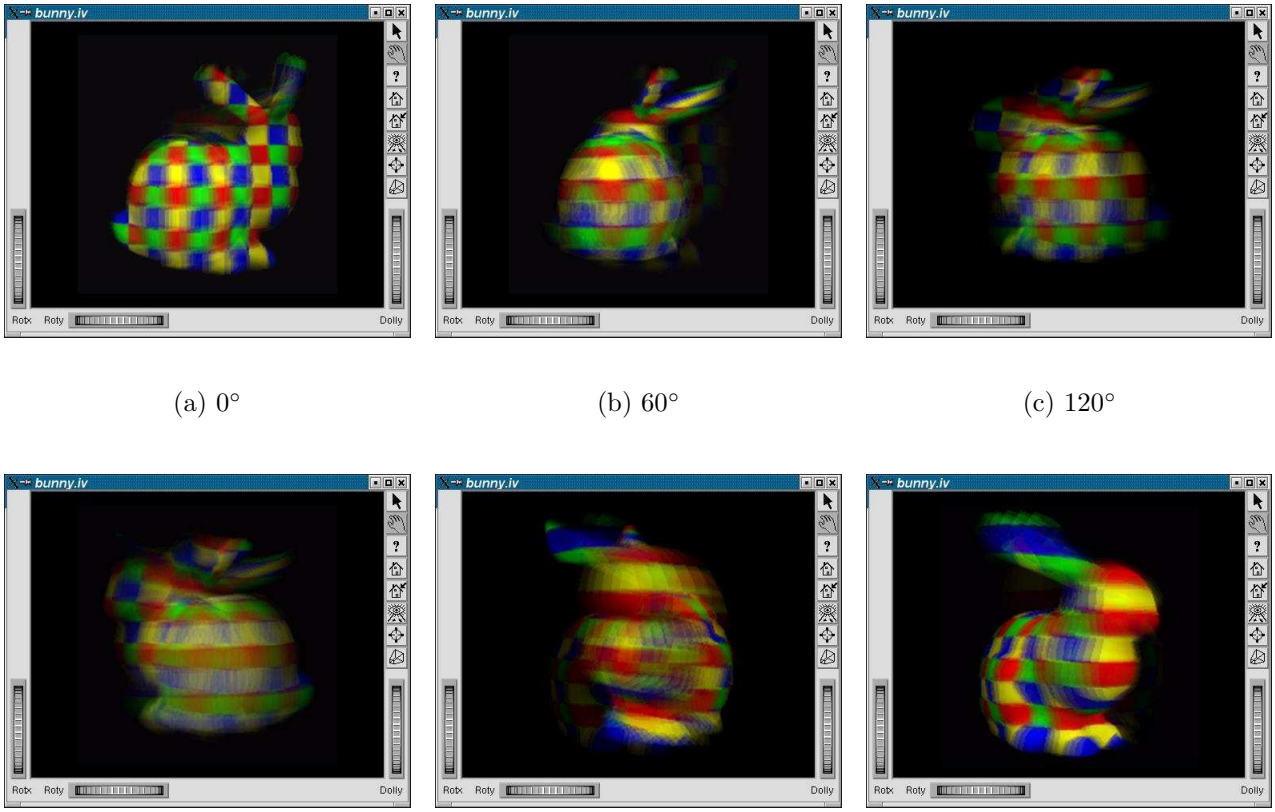
**Figure 2: Cg code for fragment shader (optimized for NVidia GeForce4 cards)**

and $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \mathbf{r}_i$ instead of the exact values $c_i$ and $\mathbf{x}_i$. The quantized output image is then computed as

$$\tilde{\mathbf{y}} = \sum_{i=0}^{M} \tilde{c}_i \tilde{\mathbf{x}}_i + \mathbf{s} = \sum_{i=0}^{M} (c_i + q_i)(\mathbf{x}_i + \mathbf{r}_i) + \mathbf{s} = \mathbf{y} + \underbrace{\sum_{i=0}^{M} (q_i \mathbf{x}_i + c_i \mathbf{r}_i)}_{A} + \underbrace{\sum_{i=0}^{M} q \mathbf{r}_i}_{B} + \mathbf{s} \approx \mathbf{y}. \quad (2)$$

The quantization errors $q_i$ and $\mathbf{r}_i$ are small compared to the values of $c_i$ and $\mathbf{x}_i$, respectively, if the available range is used appropriately. This is the case in Figures 1(a) and 1(b), and to some extent in Figures 1(c) and 1(d). Therefore the term $B$ in Equation 2 can clearly be neglected. The term $\mathbf{s}$ amounts to truncating the computation result due to the limited color resolution of the display device. Since the human eye can only resolve approximately 100 different values of each color channel within the dynamic range of a monitor, $\mathbf{s}$ does not introduce any visible error and can also be neglected. Finally, since the quantization errors are not correlated with each other, they tend to cancel out each other, making the term $A$ also negligible. We therefore conclude that quantization due to hardware limitations has no serious impact on image reconstruction. The resulting image is rendered in a plane that always faces the viewer (much like a billboard [3]).

The Cg code evaluating Equation 2 and performing the necessary range transformations is shown in Figure 2 (see [1] for full details on the Cg language). The method also works for ATI's Radeon 9700 boards. However, ATI's more flexible hardware (up to 16 floating point textures processed in a single pass) could be used to almost entirely eliminate quantization effects (except for the final result). This is left as future work.




(a) 0° (b) 60° (c) 120°

(d) 180° (e) 240° (f) 300°

**Figure 3: Bunny model reconstructed for different orientations**

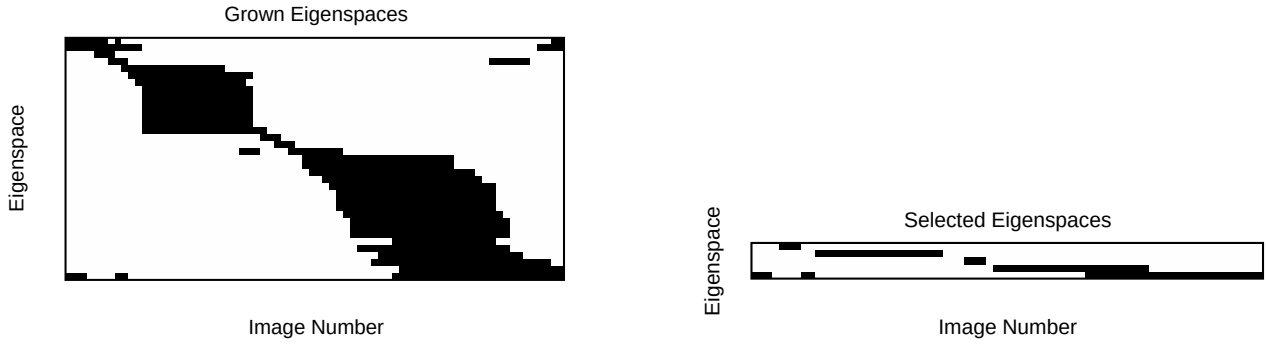
# 4 Results

We have performed experiments with several objects. Depending on the complexity of the object we get a different number of eigenspaces. For example Figure 3 shows several reconstructions of the Stanford Bunny. From 72 views used for training the algorithm created 5 eigenspaces with dimension 3 each. A high-frequency texture has been applied to the model to be able to better observe reconstruction errors.

Figure 4 illustrates eigenspace construction for the Bunny model. During the growing phase, 35 eigenspaces are taken into account (Figure 4(a)), while only 5 eigenspaces remain after the selection phase (Figure 4(b)).

## 4.1 Appearance

Some of the reconstructed images appear clean and (almost) sharp (Figures 3(a), 3(b), and 3(f)), while others are quite heavily blurred (Figure 3(d)). The reason for the blurred images is the limited number of texture units (i.e., truncating Equation 1 after a few terms).




(a) Eigenspace growing

(b) Eigenspace selection

**Figure 4: Growing and selection of eigenspaces: a black dot in the diagram indicates that this image ($x$-axis) is included in the eigenspace ($y$-axis)**

### 4.2 Performance

Since graphics hardware is highly optimized for speed, our image reconstruction method is extremely fast. Computation of the reconstruction (Equation 2) takes 560 $\mu$s for the Bunny model (Figure 3) at full resolution on a GeForce4 Ti4200. A scene consisting of 1000 instances of the eigenspace Bunny could be rendered at interactive rates (i.e., 10 to 15 frames per second).

## 5 Conclusions and future work

We presented an image reconstruction method that uses current graphics hardware to calculate the linear combination of eigenimages. The method meets the high expectations regarding rendering performance. While our test platform, the NVidia GeForce4 graphics card, fulfills the basic requirements, there is some room for improvements. The most annoying artefact is the clearly visible popping that occurs when switching between different eigenspaces. For proper blending, the system must be capable of interpolating between the eigenimages of two eigenspaces, which requires roughly twice the number of texture units. Therefore we expect these problems to be reduced when using more powerful hardware (e.g., ATI Radeon 9700). Another solution could be multipass rendering, which has not yet been implemented in this context.

Similar to other image-based rendering systems, the resolution of the final image is limited to the resolution of the input images in our method. For a close-up view of a highly detailed model it is therefore required to switch to a polygonal representation. However, as indicated in Section 4.2, our method is well suited for rendering a large number of objects at a small or medium scale.

We would also like to note that an eigenspace dimension suitable for robust image recognition might not be sufficient for generating high-quality images. This has to be taken into account when preparing eigenspaces for the use with image-based rendering methods.

Eigenspaces are computed independently for each image channel (except the fixed assignment of input images to eigenspaces). Providing a foreground/background mask in the alpha channel would improve our method's compatibility with polygonal models without increasing computation time. Moreover, the method can be put into a real-time rendering framework featuring multiresolution and progressive transmission. This involves compression of the eigenimages (e.g., using the wavelet capabilities of JPEG2000 [2]) and a budget rendering mechanism to optimally utilize available resources.

## References


[1] The NVIDIA Cg Compiler – C for Graphics. Technical Report TB-00511-001-v01, NVidia Corporation, June 2002.

[2] Michael D. Adams. The JPEG-2000 still image compression standard, December 2002. ISO/IEC JTC 1/SC 29/WG 1 N 2412.

[3] Tomas Akenine-Möller and Eric Haines. *Real-time rendering*. A K Peters, Ltd., 63 South Avenue, Natick, MA 01760, second edition, 2002. ISBN 1-56881-182-9.

[4] Daniel G. Aliaga, Thomas Funkhouser, Dimah Yanovsky, and Ingrid Carlbom. Sea of images. In *IEEE Visualization 2002*, October 2002.

[5] Dana Cobzas, Keith Yerex, and Martin Jägersand. Dynamic textures for image-based rendering of fine-scale 3D structure and animation of non-rigid motion. In George Drettakis and Hans-Peter Seidel, editors, *Proceedings Eurographics*, volume 21 of *Computer Graphics Forum*. Blackwell Publishers, September 2002. ISSN 1067-7055.

[6] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The Lumigraph. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley, August 1996. ISSN 0097-8930.

[7] Aleš Leonardis, Horst Bischof, and Jasna Maver. Multiple eigenspaces. *Pattern Recognition*, 35(11):2613–2627, November 2002.

[8] Marc Levoy and Pat Hanrahan. Light field rendering. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 31–42. ACM SIGGRAPH, Addison Wesley, August 1996. ISSN 0097-8930.

[9] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, volume 29 of *Annual Conference Series*, pages 39–46. ACM SIGGRAPH, Addison Wesley, August 1995.

[10] H. Murase and S.K. Nayar. Illumination planning for object recognition using parametric eigenspaces. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(12):1219–1227, 1994.

[11] The Web 3D Consortium. The Virtual Reality Modeling Language: International standard ISO/IEC 14772-1:1997, 1997.
URL: `http://www.vrml.org/technicalinfo/specifications/vrml97/index.htm`.