Proceedings of Mirage 2005, INRIA Rocquencourt, France, March, 1-2 2005

PanoAR: INTERACTIVE AUGMENTATION OF OMNI-DIRECTIONAL IMAGES WITH CONSISTENT LIGHTING

Thorsten Grosch

grosch@uni-koblenz.de
Institute for Computational Visualistics, University of Koblenz-Landau, Germany

ABSTRACT

Panoramic images can be used as an image-based method for navigating in a real environment. Recent publications have shown that it is possible to augment photographs with correct lighting using inverse rendering for reconstruction of lights resp. materials and differential rendering for displaying the changes. We combine these two approaches to an interactive tool with real-time modifications of the real scene maintaining consistent lighting. The possible changes are: Insertion of virtual objects with realistic lighting and shadows, insertion of new light sources by using light goniometrics and changes of materials resulting in changes in indirect lighting. The only user input is a high-dynamic-range photograph of a light probe. A 3D model is reconstructed from the light probe image using object-based reconstruction methods. The differential rendering for all user manipulations is performed using the latest generation of graphics hardware. Finally, we remove the restriction to a fixed viewpoint by projecting the environment map to the geometry in the fragment program and we present a hole-filling algorithm for the parts of the geometry which were not visible in the light probe image.

1. INTRODUCTION

Augmenting real images with correct lighting conditions becomes a more and more important area of research. Example applications are virtual furniture in a real room or new virtual buildings in a real landscape. Also, virtual characters in a movie do not look believable if the illumination is wrong. The goal of all these operations is to modify the image with changes that can not be distinguished from the real parts. This means that all changes in lighting must be calculated. Therefore, a time-consuming lighting simulation must be performed. On the other hand, the user would like to interactively modify the real photograph without a long response time. In the last few years, graphics hardware has rapidly developed and can be used to display the changes in real-time. Many approaches augment a single photograph which shows only a small part of the whole environment. We therefore decided to augment an environment map and use a panoramic image viewer to display the augmented photograph. This enables the user to look around and manipulate the whole surrounding.

2. PREVIOUS WORK

Adding virtual objects in photographs with correct lighting conditions was introduced by Nakamae et al. [29]. A method based on a radiosity simulation was developed by Fournier et al. [13]. Drettakis et al. [12] improved the update rates when moving a virtual object and Loscos et al. [27][28] separated direct and indirect light for better display quality. The first system directly using the graphics hardware for differential rendering was presented by Gibson and Murta [15], who extracted a set of directional light sources from a light probe image. The shadows of the virtual objects are displayed using shadow mapping [39] in a multi-pass algorithm. The most recent approach was presented by Gibson et al. [16]. Here, virtual objects with spatially varying light and shadows are illuminated correctly in real-time.
Using high-dynamic-range images [11] of a light probe was introduced by Debevec [10]. The light probe is a mirrored ball that captures the illumination from all incoming directions. This omni-directional image can be used for illuminating synthetic objects under real lighting conditions. Another alternative was presented by Sato et al. [34], they use a fish-eye lens to capture the illumination from the upper hemisphere in a single photograph.

An image-based system for navigating through virtual environments was introduced by Chen [6]. A set of photographs is taken from a fixed viewpoint in different directions and stitched together to a cylindrical/spherical environment texture. The user can pan and zoom the camera, but moving is only possible by switching between different panorama points or special object movies for hot spots. Yu et al. [41] showed augmented panoramic images with correctly illuminated virtual objects and new lighting conditions, calculated with a complex lighting simulation. Wong et al. [40] relighted a synthetic panorama image by precomputing basis-panorama images with a directional light source and combining the basis images.

In this paper, we present a system with more user manipulations than rotating the camera. The user can move virtual objects, add new virtual lights, change existing materials and move the camera.

The rest of the paper is organized as follows: In section 3 we explain the necessary preparations for the augmentation, section 4 describes the details about all possible user manipulations and in section 5 we conclude and give some ideas about future work.

3. PREPARATION

The first step is to take a high-dynamic-range photograph of a light probe using the algorithm described in [11]. Next, we convert the mirror ball image to a latitude/longitude environment map and use it as a texture for a sphere. We use a fragment program with a simple gamma correction as a tone mapper to display the floating point texture pixels. Usually, a high resolution image of the environment map is necessary, otherwise the resulting image looks blurry. To further improve image quality, we use a sharpening filter [23] to create a laplacian-enhanced image of the HDR environment map.

We employ the latest generation of graphics hardware because of the support of floating point textures. All calculations with input data can be done with floating-point precision and only the result is mapped to the [0,1] range. Actually, we use the 16 bit half format for our calculations because there were no visual differences to the slower 32 bit float version. At the time of development, many of the useful texture functions like filtering, cube-mapping or blending did not work with floating-point textures, so we implemented these functions in our own vertex and fragment programs. Our system is implemented using C++ with OpenGL and the RenderTexture library [24] for the floating point textures. The vertex and fragment programs are written in Cg [30].

All timing values are measured with a dual processor Xeon PC, running at 2.8 MHz, equipped with a GeForceFX 5950 graphics card. The environment map resolution is 2048 x 1024 pixels, all snapshots were made with a screen resolution of 400 x 400 pixels.

3.1. Geometry Reconstruction

The next step of our algorithm is the geometry reconstruction from a single light probe image. Reconstruction from a single image is a well-known problem: [14], [9],[37],[25],[33] present object-based algorithms for geometry reconstruction from a single image. The intrinsic parameters, especially the focal length can be determined from the position of the vanishing points in the image. With the known focal length and direction vectors for the three main axes, simple geometric primitives like rectangles or boxes can be placed in the scene using constraints like coplanarity, parallelism, orthogonality and symmetry [25]. Shum et al. [32] describe how to reconstruct geometry from a panoramic image mosaic. The extension for reconstructing geometry from a light probe image is straight forward: In our implementation, the image is first converted to the latitude/longitude environment map format (Fig. 1) and mapped as a texture onto a sphere. A camera with perspective projection is placed in the center of the sphere. An initial viewing direction is set, so that at least two vanishing points are visible. The resulting image is used as the source image for the reconstruction. When all visible objects for the initial viewing direction are reconstructed, the camera is rotated and the reconstruction can be continued using all the information already computed like the plane equations for the floor and walls. Fig. 2 shows the geometry reconstruction: The coordinate system is defined by picking parallel lines and assigning them to vanishing points. Reconstruction of the focal length is not necessary because the camera angle was set in the projection of the perspective camera. With the known focal length f and two vanishing points, the direction vectors of the world coordinate system can be calculated as [19]:

$$\vec{x} = \begin{pmatrix} -x_\infty / f \\ -y_\infty / f \\ 1 \end{pmatrix} \quad \vec{z} = \begin{pmatrix} -x'_\infty / f \\ -y'_\infty / f \\ 1 \end{pmatrix} \quad \vec{y} = \vec{z} \times \vec{x} \quad (1)$$

The user defines two points with a known distance to determine the unknown scaling factor of the geometry. Usually, we take the distance between floor and ceiling to determine the correct size of the 3D geometry. Now, simple geometric primitives can be placed in the coordinate system.
A whole room reconstructed from a light probe image is shown in Fig. 3. In this example, the reconstruction process took about 30 minutes. Problems arise from the bad resolution of the border pixels. This can be solved by taking a second photograph of the light probe from a different viewing direction as described in [10].

Figure 1. *The lat/long environment map.*

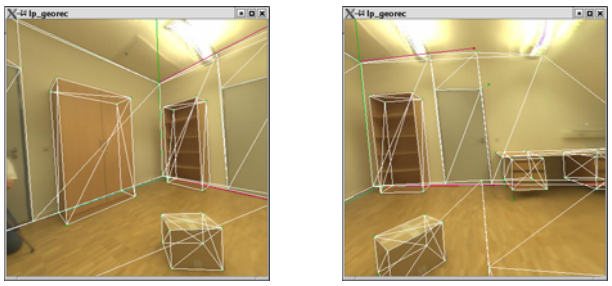

Figure 2. *Geometry reconstruction from the environment map: The left image shows the camera calibration and the reconstructed geometry, in the right image the camera is rotated and the new geometry is reconstructed using all the information from the first view*

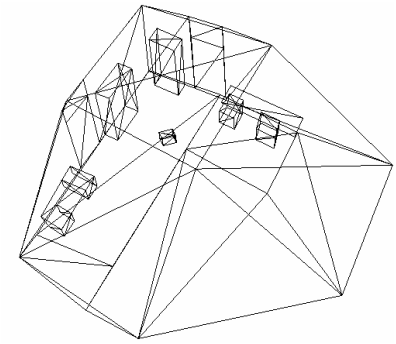

Figure 3. *The whole geometry reconstructed from the environment map image viewed from top*

3.2. Reflectance Estimation

For correct differential rendering we need to subtract radiance transfer of the light sources in the new shadow regions [16]. Therefore, we need to know the reflectance values of all geometry objects. In our implementation, all polygons are subdivided into patches. If we assume a diffuse environment, the reflectance value of a non-emitting patch is simply its radiosity divided by the irradiance

$$\rho = \frac{B}{E} \tag{2}$$

The radiosity values can be obtained from the HDR image ($B = \pi L$), where L is the average radiance of all pixels inside the patch. If we assume that all important light sources are visible in the light probe image, we can also calculate the irradiance of a receiver e :

$$E_e = \sum_s B_s F_{es} \tag{3}$$

and the reflectance value can be determined as [10]

$$\rho_e = \frac{B_e}{\sum_s B_s F_{es}} \tag{4}$$

During geometry reconstruction, we assign polygons to material groups. The reflectance value is calculated for each patch, and afterwards each material group gets the average reflectance value of all patches with the same material. To give large patches in the image a higher influence, we calculate a weighted average reflectance value. The number of pixels inside a patch is used as a weight for each patch. The reflectance value is correct for patches near the light probe position, for other positions we do not gather the complete irradiance, as shown in Fig. 4: Point B gathers irradiance from some hidden regions, not visible in the light probe image. We use a simple iteration scheme to fill the invisible regions with useful values and to improve the reflectance estimate:

1. Calculate reflectance estimates for all patches which are visible in the light probe image by using equation (4)
2. Calculate a weighted average reflectance value for each material
3. Copy the average reflectance values to patches in hidden regions
4. Calculate irradiance values E for all patches in hidden regions with equation (3) and compute a radiance estimate
5. Go back to step 1 if not converged

To speed up the computation of reflectance values, we store the irradiance values from visible regions in the first iteration (they do not change) and recompute only the irradiance from the hidden regions in the following iterations.

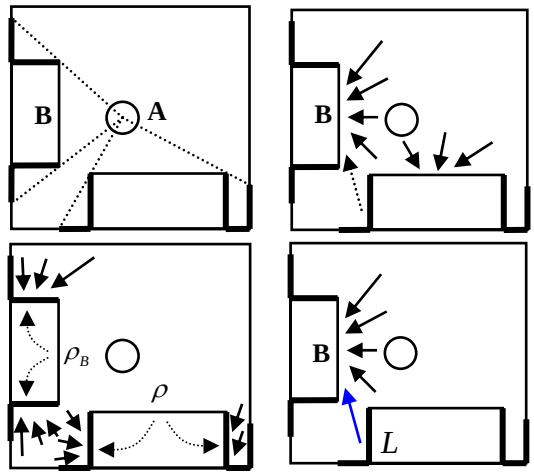

Figure 4. *The upper left image shows a simple 3D reconstruction from a light probe image. The bold parts of the geometry are not visible from the light probe position, there is no brightness information for these regions. A first reflectance estimate is calculated for the visible parts by gathering irradiance (upper right image). Note that point B gathers from a hidden region. The first reflectance estimate is copied from the visible to the invisible regions with the same material group (lower left image). The brightness in the hidden region can be estimated by gathering irradiance from all directions. With the new radiance estimate we repeat the reflectance estimation to get a better value for the irradiance. For example, point B does not gather from a dark region anymore (lower right image).*

Table 1 shows the result of the diffuse reflectance estimation for the three color channels. In our case, there are no significant changes after two iterations of our algorithm. The right column shows the real reflectance values, calculated by measuring radiance and irradiance at several locations and calculating the average of all measurements.

	1st Iteration	2nd Iteration	Real
Wall	0.98 0.95 0.94	0.91 0.88 0.86	0.79
Door	0.52 0.49 0.45	0.51 0.49 0.44	0.49
Cupboard	0.53 0.32 0.19	0.53 0.32 0.19	0.37
Floor	0.46 0.31 0.17	0.46 0.31 0.16	0.38
Table	0.57 0.43 0.28	0.53 0.39 0.25	0.38
Box	0.64 0.48 0.29	0.64 0.48 0.29	0.29

Table 1: Reflectance Values

3.3. Finding Important Light Sources

Currently, we subdivide the geometry in patches based on an area threshold. For each patch, we calculate the average radiance and then we select N patches with the highest luminous flux $\phi = BA$, like in progressive refinement radiosity [4]. This set of patches is considered as *important* light. The shadows from virtual objects are generated from these patches and the direct lighting from these patches is recomputed when the virtual objects are moved.

4. IMAGE MANIPULATIONS

4.1. Virtual Objects

For displaying virtual objects we use a modification of the method described in [16]: We first precompute an irradiance volume [21] to capture the irradiance for the whole scene. Specular materials are displayed with floating-point reflection maps. The radiance transfer from the important light sources is stored at the receiver patch vertices and interpolated. In shadow regions it is subtracted from the background. No global illumination simulation is used here, the contributions of the important light sources i = 1..N are subtracted from the real brightness values from the photograph [16]:

$$\Delta L_i = \frac{\rho}{\pi} E_i \qquad L'_{Pixel} = L_{Pixel} - \Delta L_i \qquad (5)$$

We use shadow volumes [8] to display the shadows cast by virtual objects. In contrast to shadow maps, shadow volumes have no aliasing effects at the shadow borders and the light sources are not restricted to spotlights.

When using the irradiance volume, the problem of light and shadow leaks arises: This means that shadow regions are interpolated in neighbouring regions without shadows. Therefore, we separate the *important* light from the *unimportant* light. For each grid point, two values are precomputed and stored: The first value is the normal irradiance gathered from all directions, the second value is the irradiance gathered only from unimportant sources of light. The important sources are all patches selected as described in (3.3), the unimportant sources are all remaining patches. We assume that the resulting irradiance gathered in the second value is slowly varying and the error due to linear interpolation is small. When moving the virtual object, we use two different ways to display the virtual object. The first version is fast: The virtual object is displayed by interpolating the first value of the irradiance volume. The shadows are created by building the shadow volume with all silhouette edges of the virtual object and the precomputed radiance is subtracted from the background image in the shadow regions. For a better, but more time consuming display, the second value is used and the important light is recomputed for all important light sources in the vertex program. When displaying the shadows we distinguish between shadows cast on the virtual object and shadows cast on real objects. Shadows on the virtual object are determined by creating the shadow volume with all silhouette edges of real and virtual geometry. The resulting shadow is a mask for the direct light computation in the vertex program. Shadows on real objects are determined by creating two shadow images: The first image contains the shadows of the real geometry, the second image contains the shadows of the real scene with the virtual object. When subtracting the two shadow images, we get a mask for the new shadows cast by the virtual object on real objects [15]. In these shadow regions we subtract the precomputed radiance values from the HDR background image. Fig. 5 illustrates the multipass algorithm to generate the final image (O): The new shadow caused by the virtual object is generated by creating a 0/1 image with the shadows of only real objects (A) and a shadow image with all objects (real and virtual, B). Subtracting the two images leads to an image with only the new shadows caused by the virtual object (C). This image is multiplied with the precomputed radiance values ΔL for the current light source (D), resulting in an image which contains the precomputed radiance in the shadow region (E). If we subtract this image from the background (F), the background image contains one shadow of the virtual object with correct brightness. We repeat this process for all light sources and use the generated image as an input for the next iteration (dotted arrow), so that image (F) is the background image with the previously generated shadows. For correct display of the virtual object, we use a vertex program for area lights. The virtual object is lit by the current area light (H) and this image is multiplied with the shadow image (B), resulting in an image with correct light and shadows for one area light (I). We repeat this process for all important light sources and accumulate all these images (K). Image (K) is used as an input image (J) for the next iteration. After N iterations, image (K) contains correct light and shadows from all N light sources. We add the missing light from the irradiance volume (L) to obtain the completely illuminated object (M) and insert it in front of the background by using a simple 0/1 mask of the virtual object (N).

Fig. 6 shows some images with a virtual object. In Fig.7 the simple, fast version is used, Fig. 8 shows the improved display.

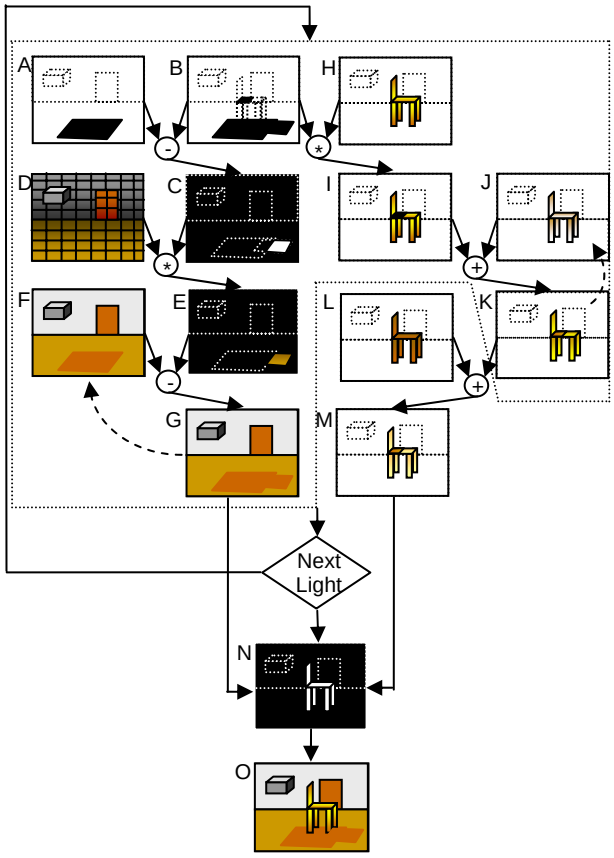

Figure 5. *The final image is created with a multipass algorithm. The steps A-K are performed in a loop for each light source (dotted region). All steps are described in section 4.1.*

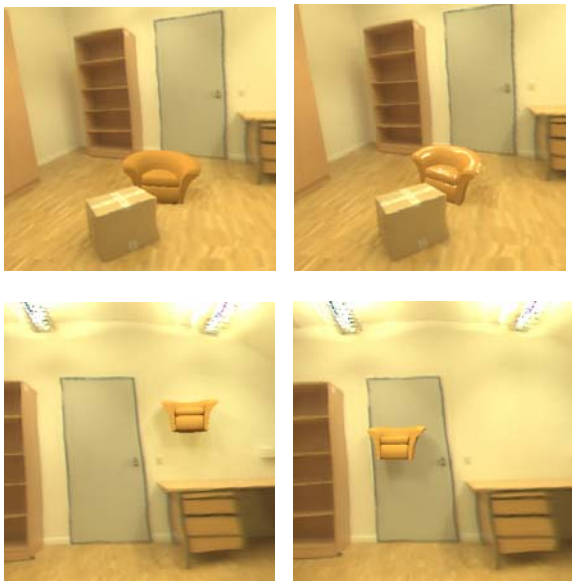

Figure 6. *A virtual armchair, illuminated by 32 important light patches, is inserted. The upper right image shows the armchair with a floating-point reflection map. The lower row shows the virtual object and the light sources. The chair can be moved at 3..20 fps (32..2 important lights)*

Anti-aliasing techniques significantly improve the realism of the virtual object as shown in [18]. We use a simple gaussian blur filter in the fragment program, applied to all virtual pixels in the composited image.

4.1.1. Moving Virtual Objects

All image manipulations can be done intuitively using the mouse pointer. For moving the virtual object, we simply align the bounding box of the virtual object with the plane of the first intersection point. The center of the virtual object is placed at the first intersection point of the ray through the pixel and moved half of the size of the bounding box along the surface normal. Fig. 9 shows some examples: If the user points to the floor, the object is aligned to the floor, if the mouse pointer points to the wall the virtual object is aligned to the wall. The object can also be rotated with the mouse and small adjustments can be done afterwards using the cursor keys.

Figure 7. *The armchair is moved under the desk at 3.2 fps with the fast display. Note the wrong shadow brightness on the floor and the missing shadow on the chair.*

Figure 8. *The armchair is moved under the desk. Note that the shadows on the armchair fit seamlessly to the real shadows. 32 important light patches are used for direct lighting and shadows. The chair can be moved at 2.6 fps with the improved display.*

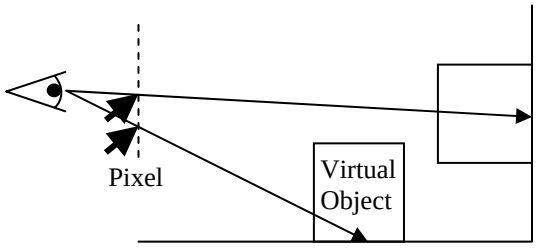

Figure 9. *Intuitive object movement: the virtual object is aligned to the geometry the user points at.*

4.2. New Light Sources

Beside virtual objects, the user can insert new light sources in the virtual environment. We use light goniometrics [35] for the description of new lights since this is one of the most realistic approximations of a real light source and better than simple point or area lights often used in computer graphics. A light goniometric describes the luminous intensity of a light source for any outgoing direction (θ, φ). Light manufacturers take such measurements of their light sources. If the data for the desired light is not available, it can be reconstructed from a set of photographs [18]. The irradiance E of a light goniometric is

$$E = \frac{I(\theta, \varphi) \cdot \cos\theta_e}{d^2} \qquad (6)$$

where d is the distance between receiver and light source and θ_e is the angle between incoming direction and surface normal. To display the direct light of a light goniometric in real-time, we convert the luminous intensity values $I(\theta, \varphi)$ to a cube map in a pre-process (Fig. 10).

The 3D position of a pixel is determined by copying the world coordinate to a texture coordinate in the vertex program, because it is interpolated for the fragment program. The fragment program calculates the light vector by subtracting the pixel position from the light position and uses the standard cube mapping equations [36] to select the corresponding pixel from one of the sides of the cube map. To handle light sources with arbitrary brightness, we use floating point textures for storing the luminous intensity values in the pixels.

The fragment program calculates the irradiance for the current position using equation (6). We calculate the shadow volume for the whole scene with virtual objects for the current light position and draw the resulting shadow in black on a white background. The shadow image is rendered in a texture and used as a mask for the radiance transfer. A mask of 1 describes a pixel which is lit, a mask value of 0 denotes a pixel in shadow. We multiply this mask with the texture containing the unoccluded light goniometric radiance to obtain light and shadows of the new light source and add it to the background image (Fig. 11). Since the reconstructed geometry is not a two-manifold, we use Bergeron's version [2] of the original shadow algorithm for correct shadows.

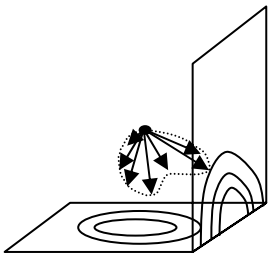

Figure 10. *A light goniometric, converted to a cube map.*

Figure 11. *A new light is inserted and can be moved and rotated interactively by the user at 20 fps. Note the shadows of real objects on real objects. The lower right image shows an artificially generated light gonimetric with different colors, stored in a single cube map.*

4.3. Changing Materials

The third possible user manipulation is the adjustment of the reflectance values. The user can point at an object and change the computed diffuse reflectance value of the corresponding material group. When changing the diffuse material from ρ to ρ', the radiance is changing from L to L':

$$L = \frac{\rho}{\pi} E \qquad L' = \frac{\rho'}{\pi} E = \frac{\rho'}{\rho} L \qquad (7)$$

The direct material changes can be visualized immediately. We simply draw the reconstructed geometry with texture coordinates set to

$$\begin{pmatrix} \rho'_R / \rho_R \\ \rho'_G / \rho_G \\ \rho'_B / \rho_B \end{pmatrix} \quad (8)$$

where ρ' is the user defined reflectance and ρ is the reconstructed reflectance value. We can not use the polygon color for the relative reflectance values because colors are always clipped to the [0,1] range. These relative reflectance values are rendered into a float texture and the HDR background image is scaled with this texture in the fragment program. To simulate color bleeding of the new surface color, we calculate delta radiosities $\Delta B = B' - B$ for all patches with a changed material and send them to all other patches. This process is similar to dynamic radiosity approaches [5][20][31]. A one-bounce approximation gives good results in most cases, but additional bounces can be computed. Due to our small reconstructed scenes, we did not use hierarchical techniques to spread the delta radiosities to different patch levels.

Figure 12. *The material of door and box are changed. The resulting color bleeding can be calculated in 0.3 - 1.0 seconds*

4.4. Camera Movement

Omni-directional images allow the user to look in every possible direction but it is not possible to move the camera. Because we reconstructed a 3D model and filled the invisible regions, we can remove this limitation by projecting the environment map onto the reconstructed geometry. This is usually done by extracting a texture for each polygon, which is rectified and mapped on the polygon [22]. We directly use the latitude/longitude image to project the background image to the geometry. The fragment program gets the interpolated 3D position as a texture coordinate, converts this direction vector in (θ, φ) coordinates and fetches the texture pixel from the environment map. For the invisible parts of the geometry we display the gathered radiance estimate as described in (3.2). This distinction is done in the fragment program by using an omni-directional depth-map storing the z-values for each direction (θ, φ) (Fig. 14). If the z-value in the texture is smaller than the real z-value in the fragment program, we use the radiance estimate. Otherwise, we take the pixel out of the environment texture. This algorithm is similar to shadow mapping. The room viewed from a novel viewpoint is shown in Fig 13, in Fig 15 all possible manipulations are combined.

Figure 13. *The camera is moved to a new position. The environment texture is projected on the geometry in the fragment program. The top of the furniture, parts of the wall and the area behind the box on the floor are not visible in the light probe image. These areas are filled with the algorithm described in (3.2)*

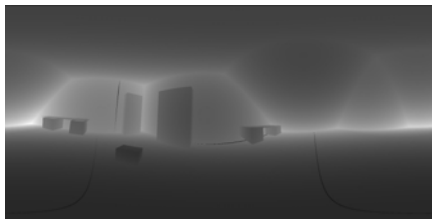

Figure 14. *Omni-directional depth map*

Figure 15. *A combination of all possible manipulations*

5. CONCLUSIONS AND FUTURE WORK

We presented a new algorithm for real-time augmentation of omni-directional images with consistent lighting. Three types of changes are possible: Addition of virtual objects, new light sources and material changes. All manipulations can be done in real-time using the latest generation of graphics hardware.
As future work we plan to improve the detection of important light sources [1]. Currently, the number of important light sources is a user parameter. This could be automated by sorting all patches by irradiance. Only the patches with the highest irradiance are selected until the sum of the remaining patches falls below a certain threshold, like in [37]. Moreover, we try to include hardware accelerated soft-shadow algorithms [26] to reduce the number of important light sources. The hole-filling algorithm can be improved by analyzing the texture in visible regions and modulating the radiance estimate with the texture in the hidden regions [28]. The hole-filling algorithm can also be used when real objects are removed or moved to a new position. Another important topic is the rapid display of changes in indirect lighting of new light sources. Here, GPU algorithms for global illumination could be used [7]. We are also interested in the correct description of the near-field of extended light sources as described in [17]. Moreover, the reconstruction of reflectance values can be extended to more sophisticated reflectance models using the method described in [3]. Finally, we would like to improve the appearance of the virtual objects by using better reflection models which can be calculated with the graphics hardware.

Acknowledgements

We would like to thank Markus Geimer and Angelika Bräunig de Colan for the corrections. Many thanks go to Oliver Abert for creating the video which can be downloaded at :
http://geri.uni-koblenz.de/Veroeffentlichungen/movies/
PanoAR.avi.

6. REFERENCES

[1] S. Agarwal, R. Ramamoorthi, S. Belongie, H. Wann-Jensen, “Structured Importance Sampling of Environment Maps”, *SIGGRAPH 2003, San Diego, CA*, TOG 22(3):605-612, July 2003

[2] P. Bergeron, “A General Version of Crow’s Shadow Volumes”, *IEEE Computer Graphics and Applications* 6, 9(Sept.) 1986, 17-28

[3] S. Boivin, A. Gagalowicz, “Image-Based Rendering of Diffuse, Specular and Glossy Surfaces From a Single Image”, *Proceedings of ACM SIGGRAPH 2001*, pp. 107-116

[4] M.F. Cohen, S.E: Chen, J.R. Wallace, D.P. Greenberg, “A Progressive Refinement Approach to Fast Radiosity Image Generation”, *Computer Graphics (SIGGRAPH 88 Proceedings)*, Vol22, No.4, August 1988, pp.75-84

[5] S.E. Chen, “Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System”, *ACM Computer Graphics (SIGGRAPH 1990 Proceedings)* Vol 24, No.4, pp.135-144, August 1990

[6] S. Chen, “Quicktime VR: An image-based approach to virtual environment navigation”, *Proceedings of SIGGRAPH 1995*, pp 29-38

[7] N. Carr, J.D. Hall, J.C. Hart, “GPU Algorithms for Radiosity and Subsurface Scattering”, *Proc. Graphics Hardware 2002*

[8] F. Crow, “Shadow Algorithms for Computer Graphics”, *Proceedings of ACM SIGGRAPH 1977*, pp. 242-248

[9] A. Criminisi, I. Reid, A. Zisserman, “Single view metrology”, *ICCV 1999*, pp. 434-441

[10] P. Debevec, “Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography”, *Proceedings of SIGGRAPH 98 (July 1998*), 189-198

[11] P.E. Debevec, J. Malik, “Recovering high dynamic range radiance maps from photographs”, *Proceedings of SIGGRAPH 97 (August 1997)*, 369-378

[12] G. Drettakis, L. Robert, S. Bougnoux, “Interactive Common Illumination for Computer Augmented Reality”, *Eighth Eurographics Workshop on Rendering (1997)*. In: Dorsey, J.; Slusallek, P. (Hrsg.): Rendering Techniques ’97. Springer Verlag, 1997, pp. 45–56.

[13] A. Fournier, A.S. Gunavan, C. Romanzin, “Common Illumination between real and computer generated scenes”, *In Proceedings of Graphics Interface ‘93 (San Francisco, CA, May 1993)*, Morgan Kaufmann, pp. 254-262

[14] S. Gibson, T. Howard, “Interactive Reconstruction of Virtual Environments from Photographs, with Application to Scene-of-Crime Analysis”, *Proc. of ACM Symposium on Virtual Reality Software and Technology 2000*, Seoul, Korea, October 2000, pp. 41-48

[15] S. Gibson, A. Murta, “Interactive Rendering with Real-World Illumination”, *Eleventh Eurographics Workshop on Rendering*. In: Péroche, B.; Rushmeier, H. (Hrsg.): Rendering Techniques 2000. Springer Verlag, 2000, pp. 365–376.

[16] S. Gibson, J. Cook, T. Howard, R. Hubbold, “Rapid Shadow Generation in Real-World Lighting Environments”, *Eurographics Symposium on Rendering 2003*

[17] M. Goesele, X. Granier, W. Heidrich, H.P. Seidel, “Accurate Light Source Acquisition and Rendering”, *ACM Transactions of Graphics (Proceedings of ACM SIGGRAPH 2003)* July 2003, Vol. 22, Number 3, pp 621-630

[18] T. Grosch, S. Mueller, W. Kresse, „Goniometric Light Reconstruction for Augmented Reality Image Synthesis“, *Graphiktag im Rahmen der GI Jahrestagung, Frankfurt am Main*, Germany, September 2003

[19] T. Grosch, „Interaktive Erweiterung und Beleuchtungsrekonstruktion von Fotos mit Radiosity“, *Diploma Thesis, Technische Universität Darmstadt 2001*

[20] D.W. George, F.X. Sillion, D.P. Greenberg, “Radiosity Redistribution for Dynamic Environments”, *IEEE Computer Graphics & Applications,* Vol.10, No.4, July 1990, pp.26-34

[21] G. Greger, P. Shirley, P.M. Hubbard, D. Greenberg, “The irradiance volume”, *IEEE Computer Graphics and Applications*, 18(2): 32-43, March 1998

[22] E. Gulliou, D. Menevaux, E. Maisel, K. Bouatouch, “Using vanishing points for camera calibration and coarse 3D reconstruction from a single image”, *In Visual Computer 16 (2000) 7*, 396-410

[23] R.C. Gonzalez, R.E. Woods, “Digital Image Processing”, *Prentice Hall 2002*

[24] www.markmark.net/misc/rendertexture.html

[25] F. van den Heuvel, “3D reconstruction from a single image using geometric constraints”, *J. Photogrammetry Remote Sensing* 53:354-368

[26] J.M. Hasenfratz, M. Lapierre, N. Holzschuch, F. Sillion, “A survey of Real-Time Soft Shadow Algorithms”, *Eurographics 2003*

[27] C. Loscos, G. Drettakis, L. Robert, “Interactive Virtual Relighting of Real Scenes”, *IEEE iTransactions on Visualization and Computer Graphics*, 6(4):289–305, 2000.

[28] C. Loscos, M.C. Frasson, G. Drettakis, B. Walter, X. Granier, P. Poulin, “Interactive Virtual Relighting and Remodeling of Real Scenes”, *Tenth Eurographics Workshop on Rendering*. In: Lischinski, D.; Ward-Larson, G. (Hrsg.): Rendering Techniques ’99. Springer Verlag, 1999, pp. 329–340.

[29] E. Nakamae, K. Harada, T. Ishizaki, T. Nishita, “A montage method: The overlaying of the computer generated images onto a background photograph”, *Computer Graphics (Proceedings of SIGGRAPH 86)* 20, 4 (August 1986), 207-214

[30] W.R. Mark, R.S. Glanville, K. Akeley, M.J. Kilgard, “Cg: A system for programming graphics hardware in a C-like language”, *Proceedings of ACM SIGGRAPH 2003*, pp. 896-907

[31] S. Müller, F. Schöffel, “Fast radiosity repropagation for interactive virtual environments using a Shadow-Form-Factor-List”, *Fifth Eurographics Workshop on Rendering (Darmstadt, Germany, May 1994)* pp.325-342

[32] H.Y. Shum, M. Han, R. Szeliski, “Interactive reconstruction of 3D models from Panoramic Mosaics”, *In Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pages 427-433, Santa Barbara, June 1998

[33] P. Sturm, S.J. Maybank, “A Method for Interactive 3D Reconstruction of Piecewise Planar Objects from Single Images”, *BMVC England*, pp. 265-274, September 1999

[34] I. Sato, Y. Sato, K. Ikeuchi, “Acquiring a radiance distribution to superimpose virtual objects onto a real scene”, *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (January - March 1999), 1 - 12

[35] H.J. Schmidt-Clausen, „Grundlagen der Lichttechnik“, *Unterlagen zur Vorlesung 2000, TU Darmstadt*

[36] D. Voorhies, J. Foran, “Reflection Vector Shading Hardware”, *ACM SIGGRAPH 1994*, 163-166

[37] G. Ward, “Adaptive Shadow Testing for Ray Tracing”, *Photorealistic Rendering in Computer Graphics*. P Brunet and F.W. Jansen, eds. Springer Verlag (EGWR91)

[38] M. Wilczkowiak, E. Boyer, P. Sturm, “Camera Calibration and 3D Reconstruction from Single Images Using Parallelepipeds”, *ICCV 2001*, pp. 142-148

[39] L. Williams, “Casting Curved Shadows on Curved Surfaces”, *Proceedings of ACM SIGGRAPH 1978*, pp. 270-274, August 1978

[40] T.T. Wong, S.H. Or, C.W. Fu, “Real-Time Relighting of Compressed Panoramas”, *Graphics Programming Methods* (edited by Jeff Lander), Charles River Media 2003, pp 375-388

[41] Y. Yu, P. Debevec, J. Malik, T. Hawkins, “Inverse global illumination : Recovering reflectance models of real scenes from photographs”, *In A.Rockford, editor, Computer Graphics (Proceedings of SIGGRAPH 99)*, Volume 19, pages 215-224. Addison Wesley Longman, August 1999