ISSN 1361 - 6161

THE UNIVERSITY
of MANCHESTER

Computer Science
University of Manchester

Interactive Rendering with Real-World Illumination

Simon Gibson and Alan Murta

Department of Computer Science

University of Manchester

Technical Report Series

UMCS-00-3-2

Interactive Rendering with Real-World Illumination*

Simon Gibson and Alan Murta

Advanced Interfaces Group
Department of Computer Science
University of Manchester
Oxford Road, Manchester, UK.
sg@cs.man.ac.uk, amurta@cs.man.ac.uk

March 16th 2000

Recent technical reports issued by the Department of Computer Science, Manchester University, are available by anonymous ftp from ftp.cs.man.ac.uk in the directory pub/TR. The files are stored as PostScript, in compressed form, with the report number as filename. They can also be obtained on WWW via
URL http://www.cs.man.ac.uk/cstechrep/index.html. Alternatively, all reports are available by post from The Computer Library, Department of Computer Science, The University, Oxford Road, Manchester M13 9PL, UK.

*Refereed by: Toby Howard

Abstract

We propose solutions for seamlessly integrating synthetic objects into background photographs at interactive rates. Recently developed image-based methods are used to capture real-world illumination, and sphere-mapping is used illuminate and render the synthetic objects. We present a new procedure for approximating shadows cast by the real-world illumination using standard hardware-based shadow mapping, and a novel image composition algorithm that uses frame-buffer hardware to correctly overlay the synthetic objects and their shadows onto the background image. We show results of an OpenGL implementation of the algorithm that is capable of rendering complex synthetic objects and their shadows at rates of up to 10 frames per second on an SGI Onyx2.

1 Introduction

The seamless integration of synthetic objects into real photographs or video images is a common requirement in fields such as augmented reality and digital visual effects. Achieving a high degree of realism in such applications requires first matching the geometric characteristics of both the synthetic and real cameras, and then shading the synthetic objects so that they appear to be illuminated by the same lights as the other objects in the background image. By rendering the synthetic objects with the same illumination, important visual cues are provided that increase the apparent accuracy of the integration, and aid the user's judgement of spatial layout.

In the field of augmented reality, many researchers have focussed on the first of these problems, namely camera calibration and registration (for example, see [22, 1]). Few methods, however, have been proposed for solving the problem of illuminating the synthetic objects in a realistic and believable way. This is mainly due to the enormous complexities of real-world illumination, which includes both direct and indirect illumination from complex light-sources, shadows, and glossy reflections and refractions. Global illumination algorithms (e.g. [14]) simulate these effects using complex and time consuming approaches that are not suitable for interactive augmented reality applications.

In this paper, we provide solutions for illuminating and integrating synthetic objects into real-world images at interactive rates. Our method combines the image-based capture and rendering of real-world illumination with novel shadow generation and image composition algorithms that uses computer graphics hardware to generate photo-realistic images containing both real and synthetic objects.

2 Background and Related Work

One of the earliest algorithms for compositing synthetic objects into background images was proposed by Nakamae *et al.* [17]. Their solution involved estimating the location of the sun and levels of ambient light to illuminate synthetic architectural models rendered onto background photographs.

Fournier *et al.* [9] proposed using an approximate model of the scene to generate estimates of diffuse reflectance characteristics, and radiosity algorithms [10] to illuminate synthetic objects. Using this approach, a re-calculation of the radiosity solution must be performed whenever any form of interaction with the synthetic objects occurs.

Radiosity algorithms were also employed by Drettakis *et al.* [8], who proposed a solution for providing common illumination between real and synthetic objects in an interactive setting. Their approach extended the work of Fournier *et al.* by employing a semi-automatic algorithm for constructing the model of the environment, and an incremental radiosity algorithm for the rapid update of illumination for moving objects.

One of the disadvantages with the approaches described above is that they require a model of the entire environment in order to illuminate the synthetic objects. In many situations (for example, a complex outdoor scene), it is both difficult and time-consuming to construct such a model. In [5], Debevec proposed a different approach to capturing real-world lighting. Instead of storing illumination details on a model of the scene, he captured an omni-directional image (or *radiance-map*) that represented the incident illumination at a single point in the scene. This radiance-map was then used to construct a very simple light-based model of the parts of the scene which were distant from the synthetic objects. An approximate, material-based model of the environment near the synthetic objects was then built, and the synthetic objects were composited into the background image using the RADIANCE global illumination system [26] and a differential rendering algorithm [5]. A similar algorithm has also been proposed by Sato *et al.* [20], who use a hemispherical fish-eye camera lens and stereoscopic reconstruction algorithm to capture illumination and model the distant parts of the environment respectively.

Debevec and Malik [6] have also addressed the problem of representing the dynamic range of real-world lighting conditions. By capturing images at different levels of exposure, the response function of the imaging system may be recovered, and the images combined into a single high-dynamic range photograph.

All of these algorithms use expensive global illumination techniques to illuminate the synthetic objects. With the exception of Drettakis *et al.* [8], these techniques are not applicable when any form of interaction is required. The incremental algorithm employed by Drettakis was reported to provide updates on the order of two to three seconds per frame for a simple environment, but is only capable of modelling diffuse illumination, and so cannot accommodate the wide variety of reflection characteristics we see in real-world images.

As software-based approaches to realistic illumination have progressed, researchers have also started to examine ways in which standard computer graphics hardware can be used to perform complex lighting calculations that go beyond the original Gouraud shaded, texture mapped triangles the hardware was intended to produce. Walter *et al.* [25] proposed using hardware-based lighting calculations to approximate reflections from glossy surfaces during an interactive walk-through of a pre-computed global illumination solution. A multi-pass algorithm was also developed by Diefenbach and Badler [7] that used the rendering pipeline to provide high-quality images containing shadows, reflections and refractions.

More recently, Heidrich and Seidel [13] published algorithms for performing realistic shading calculations using computer graphics hardware. They proposed using multi-pass methods for high-quality local illumination with physically-based reflection models, as well as techniques for pre-integrating omni-directional images in order to render objects with complex material characteristics using *sphere-mapping* [2, 11]. In parallel, Cabral *et al.* [4] also developed similar techniques. They pre-integrated an omni-directional image with a surface BRDF to form a sphere-map.

3 Overview

In this paper, we present an algorithm that enables complex synthetic objects to be realistically illuminated and seamlessly combined with background images at interactive rates. Currently, our implementation is restricted to rendering images from a single viewpoint, but we briefly discuss possible extensions at the end of the paper.

Like Debevec [5] and Sato *et al.* [20], we partition the environment into three sections: the *synthetic objects* are the objects we wish to add into the image, the *local scene* is the part of the environment that the local objects will interact with[1], and the *distant scene* comprises the parts of the environment considered to be far enough away that the effect the synthetic objects have on them is negligible.

Instead of employing expensive global illumination algorithms, we utilise recent advances in hardware-based rendering [4, 13] to model the way the distant scene illuminates the synthetic objects. The illumination in an environment is first captured in the form of an omni-directional image using a camera, and then a series of high-dynamic range *basis radiance-maps* are pre-computed. Combinations of these maps are used at run-time to approximate the illumination for different synthetic materials. We currently ignore inter-reflection effects between the synthetic objects, as well as reflections of the synthetic objects in the local scene.

We also introduce a new approach for modelling the shadows that the synthetic objects cast onto the local scene (and vice versa) as well as self-shadowing effects on the synthetic objects. By analysing the omni-directional image captured by the camera, we automatically position a number of representative directional light-sources around the synthetic objects so that the accumulated effect of the shadows from each light-source approximates those cast by the real-world illumination. We also present a novel image composition algorithm that is capable of merging both the synthetic objects and shadows into the background image at interactive rates.

The remainder of this paper is organized as follows. In the next section we outline how the omni-directional images that store illumination are captured. Following that, we describe how sets of basis radiance-maps are generated in Section 5. Our image composition algorithm is then discussed in detail in Section 6, and Section 7 presents our methods for selecting the position and intensity of the representative light-sources.

[1] Currently, a geometric model of the local scene is constructed by hand, although we are developing automated techniques that will be presented in a future paper.

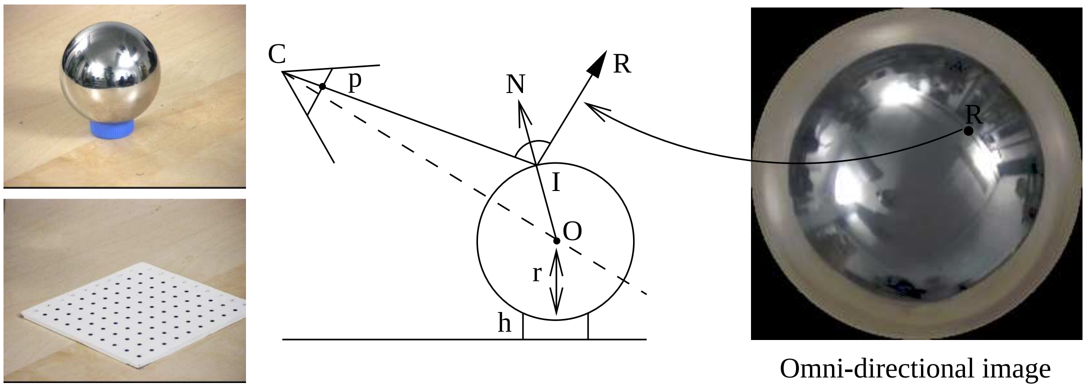

Figure 1: A photograph of the metallic sphere, and calibration pattern (left), geometry for the image-based warp (middle), used to generate an aligned omni-directional image (right).

Finally, we give results in Section 8 and describe areas for future research in Section 9.

4 Illumination and Image Acquisition

As mentioned above, we utilise recent advances in the image-based capture and storage of illumination [6, 5] to correctly illuminate synthetic objects. Currently, we only capture illumination at a single position in the scene, and assume that this is a good approximation of the actual light illuminating each synthetic object.

The acquisition process begins with the capture of the background image and camera position. This is achieved using a simple calibration pattern consisting of a 9x9 grid of small dots and a standard camera calibration algorithm [24].

Following that, we must construct a high-dynamic range omni-directional image. This is achieved by first capturing an image of a smaller calibration grid, which has been positioned such that its axes are aligned with the larger grid used to calibrate the background image. Once the camera is calibrated (again, using Tsai's calibration algorithm [24]), we position a small metallic sphere within the frame, and capture images at multiple exposure levels (Figure 1, left).

The algorithm proposed by Debevec [6] is then used to combine the set of captured images into a single high-dynamic range (HDR) photograph. In order to construct a usable omni-directional image, we must then warp the HDR image to a pre-determined orientation. For the sake of convenience, we orient the omni-directional image vertically, as shown on the right of Figure 1. In this orientation, a vector through the centre pixel of the image corresponds to the +Y (vertical) axis, and the circumference of the circle represents the -Y axis.

In order to warp the HDR image, we must calibrate the position of the metallic sphere (Figure 1, centre). The sphere is of known radius r, so its position is found simply by measuring the height it is mounted above the calibration grid, h, and then identifying to sub-pixel accuracy the pixel in the image in which the centre of the camera lens is visible. A line from the centre of projection, C, through this pixel will also pass through the centre of the sphere, O, which can be found using simple algebra.

Once the position of the sphere has been determined, we can proceed to warp the HDR image to the correct orientation. For each pixel in the aligned omni-directional image, we calculate the pixel's direction vector R. Knowing this vector, along with the sphere centre and radius, we can find the corresponding normal vector N and sphere surface position I. With knowledge of the intrinsic and extrinsic camera parameters from the calibration, this in turn allows the corresponding pixel in the HDR image, p, to be found, and the correct radiance value stored in the omni-directional image. We also generate another HDR image of the metallic sphere from a different camera position, and warp it to the same orientation. By merging both oriented omni-directional images, we may remove the image of the camera, and errors due to the singularity of the sphere horizon locus in the image warp.

5 Basis Radiance-Maps

Once an omni-directional radiance-map has been captured, it must be manipulated into a form suitable for rendering with computer graphics hardware. As each synthetic object is drawn, we use hardware-accelerated sphere-mapping [2, 11] to approximate the appearance of the object under the captured illumination conditions.

In [4, 13], it was proposed that an omni-directional radiance-map could be pre-integrated with a BRDF (bi-directional reflectance distribution function) to generate a sphere-map that stores the outgoing radiance for each surface normal direction. This pre-integration, which needs to be performed for each different BRDF, is too slow whenever an application requires the interactive manipulation of materials[2].

As an alternative approach, we propose that a small number of *basis radiance-maps* (storing incoming radiance) may be pre-computed for a specific local reflection model. At run-time, we combine these basis maps according to the particular reflection model coefficients associated with each material, and generate the appropriate sphere-map. Whenever a material changes, or a new one is introduced, all that is required is that the appropriate basis maps are re-combined according to the new coefficients.

In the examples presented in this paper, we have used the modified Phong reflectance model [16]. With this model, the reflected radiance in direction ω_r is separated into diffuse and specular components. For the diffuse component, ω_r corresponds to the surface normal, and we integrate over the hemi-sphere, Ω_r, defined by the surface normal to obtain irradiance, $E(\omega_r)$

$$E(\omega_r) = \frac{1}{\pi} \int_{\Omega_r} L_i(\omega_i)(\omega_i \cdot \omega_r) d\omega_i \tag{1}$$

where $L_i(\omega_i)$ is the radiance in the omni-directional image for direction ω_i. The specular basis maps are obtained by integrating with different values of the surface roughness

[2]On a single 250MHz R10000 MIPS processor, we have found that the time required to integrate a 256x256 radiance-map is approximately 4 minutes.

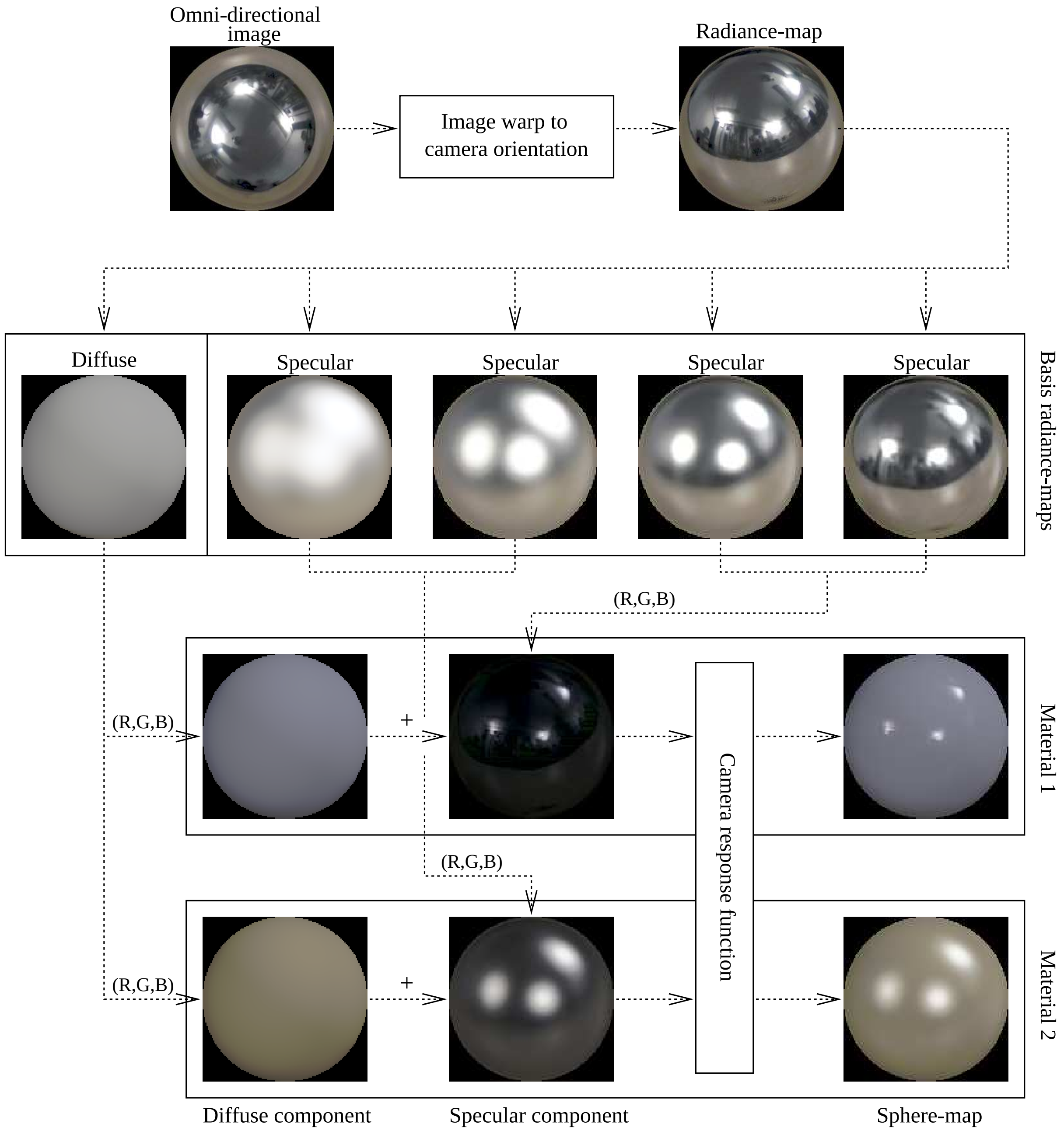

Figure 2: The stages required to generate a correctly oriented sphere-map for each material. Radiance-maps representing specular components for different surface roughness parameters are combined with a diffuse radiance-map. The camera-response function is then applied to generate a suitable sphere-map.

parameter, ψ, with Ω_r correcponding to the sphere of reflected directions, ω_r

$$L_\psi(\omega_r) = \frac{\psi + 2}{2\pi} \int_{\Omega_r} L_i(\omega_i) \max(0, \omega_r \cdot \omega_i)^\psi d\omega_i \tag{2}$$

We illustrate the combination process in Figure 2, where a typical set of basis radiance-maps is shown. The map on the left corresponds to the diffuse component, and the maps on the right correspond to the specular components for different amounts of surface roughness. Underneath these basis maps, we show how they are combined to generate sphere-maps for two different materials. The surface roughness parameter for each material is used to identify a specular basis map, which is combined with the diffuse map according to the reflection coefficients of the material. Since it is unlikely that the surface roughness parameter will match exactly one of the specular basis maps, we select the two maps having the most similar roughness parameter, and linearly interpolate between them.

After the specular and diffuse maps have been combined together, we are left with a high-dynamic range radiance-map that represents the radiance reflected back towards the viewer for every direction facing the camera. The final stage of sphere-map generation is the application of the camera response function [6] to translate the radiance-map into a texture map that can be utilised by computer graphics hardware.

6 Image and Shadow Composition

In this section, we present a multi-pass algorithm that uses sphere-mapping [2, 11], shadow-maps [19, 21] and accumulation-buffer hardware [12] to overlay the synthetic objects and their associated shadows on the background image.

It is important that the synthetic objects cast believable shadows onto the local scene in order to improve a user's perception of spatial layout [23]. The synthetic objects should also be shadowed by the objects in the local scene, and the local scene must not cast shadows onto itself since these are already present in the background image. Self-shadowing is, however, required on the synthetic objects. We must take care that this self shadowing only occurs on parts of the objects that are facing *towards* the source of illumination, because the reduction in radiance on back facing sections has already been accounted for within that object's sphere-map.

We approximate shadows cast by the real-world illumination using a small number of *representative directional light-sources*. Rather than considering these as sources of light, we will instead consider them as sources of shadow, because they will be used to subtract light from the already illuminated background image. Each 'shadow-source', i, has associated with it a direction ω_i, a colour I_{i_λ} where $\lambda = (R, G, B)$, and a *shadow ambient* factor [15] α_i, where $(0 < \alpha_i < 1)$. These control the position, colour and darkness of its shadows respectively. In Section 7 we will discuss how these parameters may be chosen so as to best approximate the shadows cast by the real-world illumination.

The generation of a single frame begins with the shadows cast by each shadow source being summed into the accumulation buffer. Once all shadow-sources have been pro-

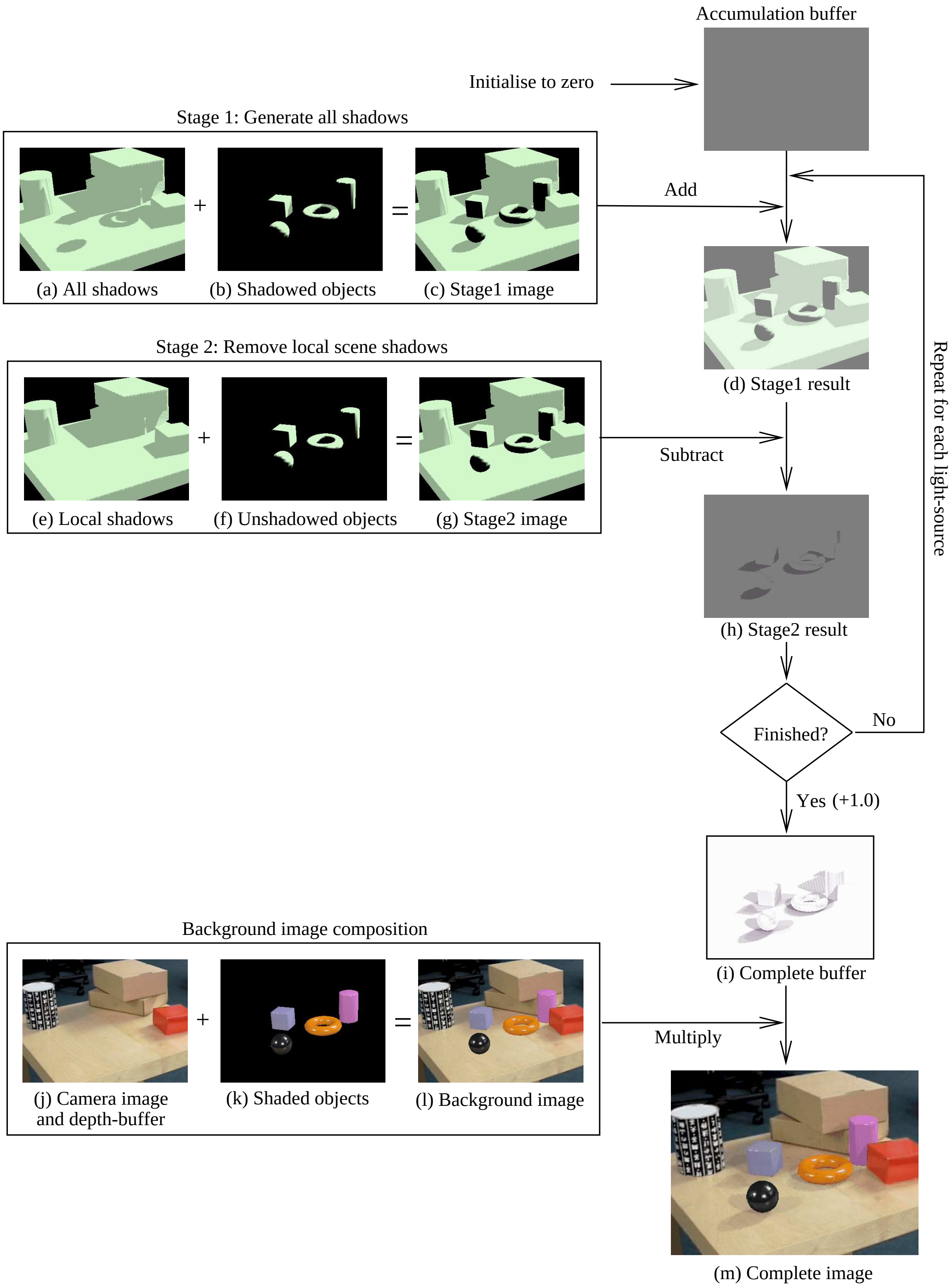

Figure 3: The image and shadow composition process (see text for explanation).

cessed, the contents of the accumulation buffer are blended with the background image and synthetic objects. This composition process is illustrated diagrammatically in Figure 3, and is implemented entirely using hardware-accelerated OpenGL features.

Due to the restrictions of shadow-mapping, the shadows cast by a single source must be generated in two stages: firstly, we generate all the shadows in the scene (i.e. those cast by the synthetic objects onto themselves and the local scene, and the shadows cast by the local scene onto itself and the synthetic objects). Following that, we remove the shadows cast by the local scene onto itself.

6.1 Stage One: Addition of All Shadows

Stage one of the shadow generation process for a single source begins with the construction of a *shadow-map* [19, 21]. This is produced by rendering the local scene and synthetic objects into the depth-buffer, from the point of view of the shadow-source. The local scene is then drawn from the camera viewpoint, with the shadow-map activated and lighting turned off. All local scene triangles are assigned the colour of the shadow-source, meaning that areas in shadow are rendered with a colour $\alpha_i I_{i_\lambda}$, and those not in shadow with a colour I_{i_λ} (Figure 3a). Note that we have artificially reduced the value of α_i in these images in order to ensure the shadows are more easily identifiable.

The synthetic objects are then be drawn, again using the shadow-map. If this is done in the same way as for the local scene, however, self-shadowing artifacts on back-facing triangles will be present in the image (these are unwanted for the reasons outlined above). To avoid these, we render the synthetic objects so that only triangles facing towards the source have shadows cast upon them. This is achieved by placing an OpenGL light-source the same direction as the shadow-source, and rendering each synthetic object with a material that has ambient and diffuse components of zero, a specular component of I_{i_λ}, and a Phong exponent of zero. This ensures that triangles facing away from the shadow source are rendered in black, and so have no shadows cast upon them (Figure 3b). The resulting image (Figure 3c) is then added to the current contents of the accumulation buffer, as shown in Figure 3d[3].

6.2 Stage Two: Removal of Local Scene Shadows

Stage two of the shadow generation process is required to remove shadows cast by the local scene onto itself. Again, a shadow-map is constructed, but this time only using the triangles in the local scene. The local scene is then rendered from the point of view of the camera, using this shadow-map and a colour of $\alpha_i I_{i_\lambda}$ (Figure 3e). Shadowing is then disabled, and the synthetic objects are drawn so that those triangles facing away from the shadow-source are black (Figure 3f). The resulting image is then subtracted[4] from

[3]Note that the contents of accumulation buffer can take on values in the range [-1:1]. We have therefore scaled the buffer contents to the range [0:1] for illustration purposes in Figures 3d and h.

[4]Since data may only be added into the accumulation buffer, this subtraction is in fact achieved by scaling the image by -1.0 and adding.

the accumulation buffer, resulting in an image similar to that shown in Figure 3h.

The overall effect of this sequence of operations is to add $I_{i_\lambda}(\alpha_i - 1)$ to regions in the accumulation buffer that are occluded from shadow-source i. Regions that are not in shadow remain unaffected. Shadow generation may be continued for any number of additional shadow-sources simply by repeating the two-stage process outlined above, using the current contents of the accumulation buffer as the starting point for the next source. Once all shadow-sources have been processed, we add 1 to each pixel in the accumulation buffer, resulting in an image similar to that shown in Figure 3i. If we denote the occlusion of a pixel x to source i as $g(x, i)$, where $g(x, i)$ is 1 if the pixel is shadowed, and 0 otherwise, then the contents of the accumulation buffer after M sources, and the final addition, will be:

$$1 + \sum_{i=1}^{M} I_{i_\lambda}(\alpha_i - 1) g(x, i) \tag{3}$$

Pixels not shadowed from any source will have a value of 1, and pixels in shadow will have values in the range $[0 : 1]$, depending upon the number and intensities of the occluded sources.

The last stage of the image composition process is the construction of the background image. This is achieved by rendering the camera image into the colour buffer, and the local scene geometry into the depth buffer (Figure 3j). The synthetic objects are then drawn, using their appropriate sphere-maps. Finally, this background image is multiplied by the contents of the accumulation buffer (using standard OpenGL blending operations), and the result (Figure 3m) is displayed on screen.

7 Shadow-Source Positioning

In this section, we discuss how suitable directions, colours and shadow ambient factors may be determined for any number of shadow-sources. In [5], shadows cast by the synthetic objects were generated using time-consuming ray-tracing techniques [26]. We use similar techniques to generate a reference image of the shadows cast by a simple object (e.g. a sphere) onto a diffuse horizontal plane on which the object is resting. A multi-dimensional optimisation is then performed, which selects shadow-source parameters giving the best fit to the target reference image.

The reference image is generated by calculating an occluded radiance estimate, $L_o(x)$, for each pixel x

$$L_o(x) = \rho \int_\Omega L(\omega)(1 - g(x, \omega))(N \cdot \omega) d\omega \tag{4}$$

and an unoccluded radiance estimate, L_u (which is independent of pixel position),

$$L_u = \rho \int_\Omega L(\omega)(N \cdot \omega) d\omega \tag{5}$$

where ρ is the diffuse reflectance of the horizontal plane (we assume a constant value of 0.5), Ω is the upper hemisphere of the omni-directional image, $L(\omega)$ is the radiance stored in direction ω, N is the normal of the horizontal plane, and $g(x, \omega)$ is 1 if the ray from x in direction ω is occluded by the sphere, and 0 otherwise.

We then determine a *reference weighting factor*, $\tau_\lambda(x)$ that gives the ratio of the occluded and unoccluded pixel intensities for channel λ:

$$\tau_\lambda(x) = \frac{R_\lambda(L_o(x), \epsilon)}{R_\lambda(L_u, \epsilon)} \tag{6}$$

where $R_\lambda(L, \epsilon)$ is the camera response function [6] for channel λ, which maps radiance L to a pixel intensity, assuming an exposure of ϵ.

Assume we want to position M shadow-sources, each of which has an associated direction ω_i, colour I_{i_λ}, and shadow ambient factor α_i. The choice of M directly affects the frame time, and so should be kept small enough to maintain interactivity (we currently use between four and eight sources, depending upon the contents of the omni-directional image). From Equation 3, the accumulated effect of these M shadow-sources will be to multiply a background pixel by an *approximate weighting factor*, $\sigma_\lambda(x)$, where

$$\sigma_\lambda(x) = 1 + \sum_{i=1}^{M} I_{i_\lambda}(\alpha_i - 1) g(x, \omega_i) \tag{7}$$

The shadow-source parameters must therefore be chosen such that

$$\sigma_\lambda(x) \approx \tau_\lambda(x) \tag{8}$$

for all x in the reference image. We propose that this may be cast as a multi-dimensional optimisation problem, which attempts to minimise the RMS difference between the image of reference weighting factors, $\tau(x)$, and the image of approximate weighting factors, $\sigma(x)$. Rather than employing complex global minimisation algorithms such as Simulated Annealing [18] to solve this problem, we have found that reliable and repeatable results may be obtained using the much simpler Downhill Simplex method [18] and a bounded linear least-squares solver [3], in conjunction with a stratification of the problem domain. For source i, we will derive a direction ω_i and quantity C_{i_λ}, where

$$C_{i_\lambda} = I_{i_\lambda}(\alpha_i - 1) \tag{9}$$

From C_{i_λ}, the colour and ambient shadow factor of each source can then be found

$$\begin{aligned} I_{i_\lambda} &= C_{i_\lambda} / m_i \\ \alpha_i &= 1 - m_i \end{aligned} \tag{10}$$

where $m_i = \max_\lambda(C_{i_\lambda})$. The optimisation process for each shadow-source is split into two phases: the first phase chooses a direction ω_i and the intensity (luminance) of C_{i_λ} using the Downhill Simplex method. The second phase then determines the best values for C_{i_λ} for $\lambda = (R, G, B)$ using a bounded linear least-squares solver.

The optimisation process is started by positioning a single shadow-source in a random direction ω_1, and assigning $C_{\lambda_1} = 1/(2M)$ for all λ (we have found this to be a useful

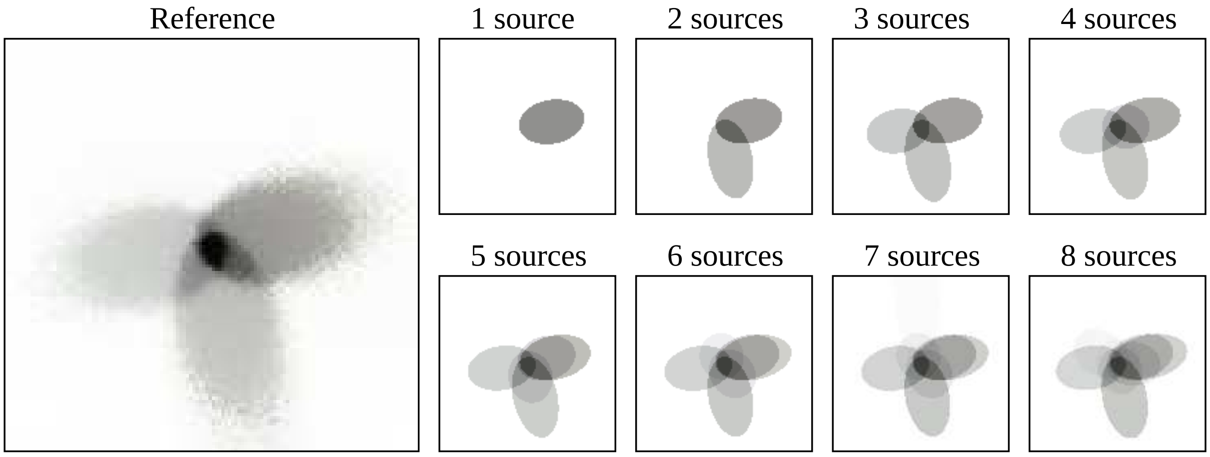

Figure 4: Shadow approximation using representative sources. An optimisation procedure positions M point shadow-sources so as to best fit the reference image, shown on the left. Results are given on the right for between one and eight sources.

initial guess for the shadow intensity). The Downhill Simplex algorithm is then run to find the parameters for this single shadow-source that provide the best fit of $\sigma_\lambda(x)$ to the reference weighting factors, $\tau_\lambda(x)$ for all x.

Once the direction ω_1 and intensity of C_{λ_1} have been chosen, we determine the optimal values of C_{i_λ} for all points x_i $(i = 1 \ldots N)$ in the reference image, and all m sources positioned so far (i.e. $m = 1$ initially) by solving

$$\text{minimize } \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \quad \text{subject to } (-1 < C_{i_\lambda} < 0) \tag{11}$$

where

$$\mathbf{A} = \begin{bmatrix} g(x_1, \omega_1), & g(x_1, \omega_2), & \cdots & g(x_1, \omega_m) \\ g(x_2, \omega_1), & g(x_2, \omega_2), & \cdots & g(x_2, \omega_m) \\ \vdots & \vdots & \ddots & \vdots \\ g(x_N, \omega_1), & g(x_N, \omega_2), & \cdots & g(x_N, \omega_m) \end{bmatrix}$$

$$\mathbf{x} = [C_{\lambda_1}, C_{\lambda_2}, \ldots, C_{\lambda_m}]^T$$

$$\mathbf{b} = [\tau_\lambda(x_1) - 1, \tau_\lambda(x_2) - 1, \ldots, \tau_\lambda(x_N) - 1]^T \tag{12}$$

Once the values of C_{λ_1} have been chosen, we add a second shadow-source (again, with random direction and $C_{\lambda_2} = 1/(2M)$), but use the results of the first optimisation as the new starting conditions for the first shadow-source. The Downhill Simplex algorithm is repeated, and Equation 11 solved for $m = 2$. This process is repeated until all M shadow-sources have been positioned, each time using the results of the previous optimisation, along with one random direction ω_m and $C_{\lambda_m} = 1/(2M)$, as the new initial estimates.

Figure 4 shows the results of this optimisation process for the omni-directional image used throughout the paper. On the left-hand side we show the reference weighting factors, $\tau_\lambda(x)$ for all pixels x. The approximate factors, $\sigma_\lambda(x)$, for the sequence of optimisations $1 \leq m \leq M$ that result from positioning 8 shadow-sources are shown on the right.

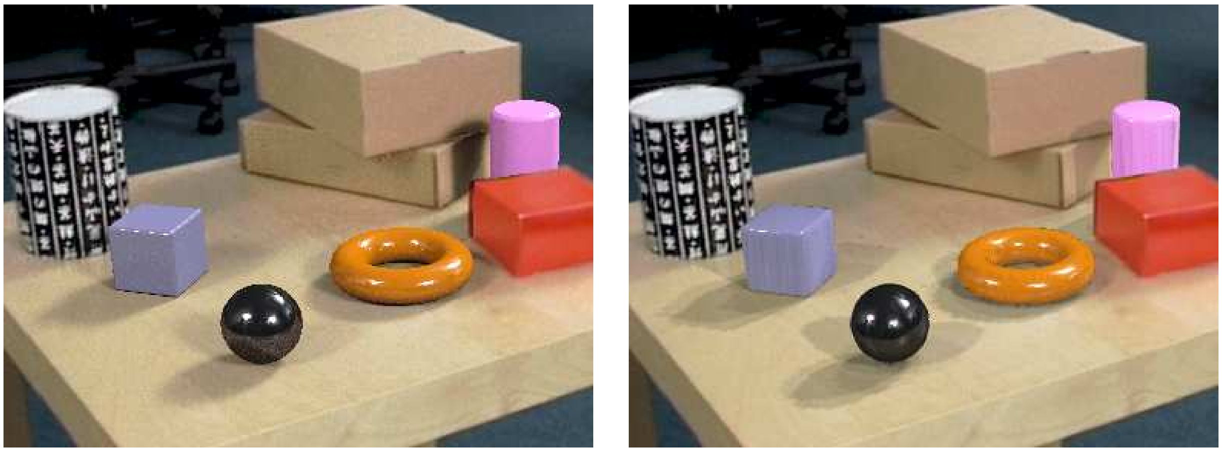

Figure 5: Comparison of ray-traced (left) and hardware generated images (right). The ray-traced image was generated using the RADIANCE lighting-simulation software [26] in approximately 2 hours. The right-hand image was rendered at almost 10 frames-per-second.

Overall, the time required to perform this sequence of optimisations was approximately two minutes on a single 250MHz R10000 MIPS processor. Although we have no direct proof that these results represent the true global minimum of the problem, we have found that they are repeatable and, importantly, independent of the initial random positioning of each source. Also, since the time required to render each frame is directly affected by the number of shadow-sources, we are able to use the intermediate results for fewer sources if the frame-time with all M becomes too great.

8 Results

We have implemented this algorithm in OpenGL on a Silicon Graphics Onyx2 workstation with a single InfiniteReality2 rendering pipeline and one 250MHz MIPS R10000 processor. Figure 5 gives a side-by-side comparison of a ray-traced image and one generated in real-time using the techniques presented here. The ray-traced image was produced using the RADIANCE global illumination software [26], and differential rendering techniques similar to those developed by Debevec [5]. The total time required to produce this image was approximately 2 hours. The image on the right was rendered at a sustained rate of almost 10 frames-per-second (fps) on the Onyx2. Eight shadow-sources were used to generate the shadows in this image. It is clear from this comparison that the real-time shadows are a good match for those obtained using expensive ray-tracing algorithms, although differences do exist. Most noticeably, the shadow cast by the cylinder onto a vertical side of the box is not as dark as it should be. This is due to the fact that the shadow intensities are calculated for a horizontal plane with its normal facing upwards, and will differ for vertical surfaces. Figure 6 compares real-time self-shadowing effects against those produced using ray-tracing. The synthetic object in this image contains

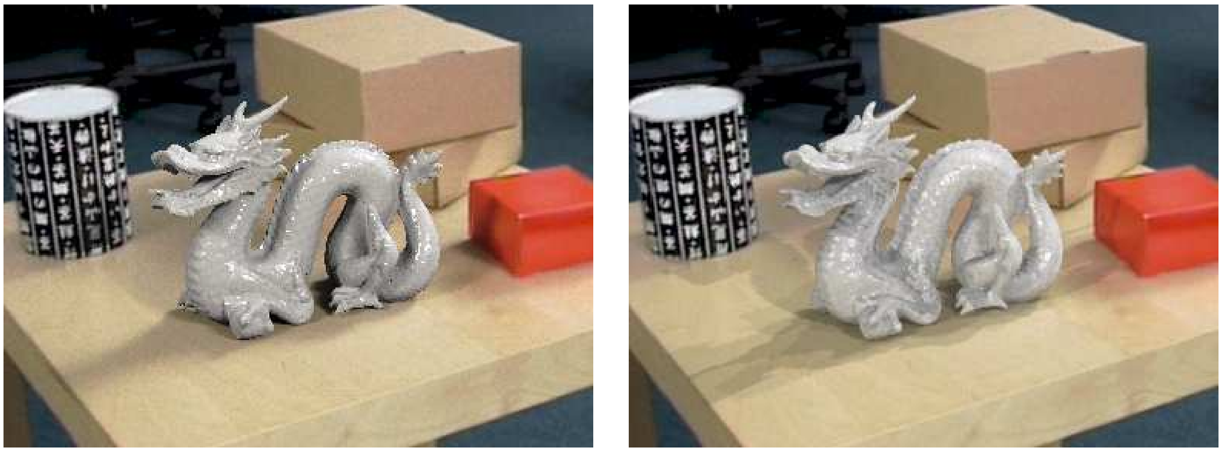

Figure 6: Comparison of self-shadowing produced using ray-tracing (left) and hardware-based rendering (right).

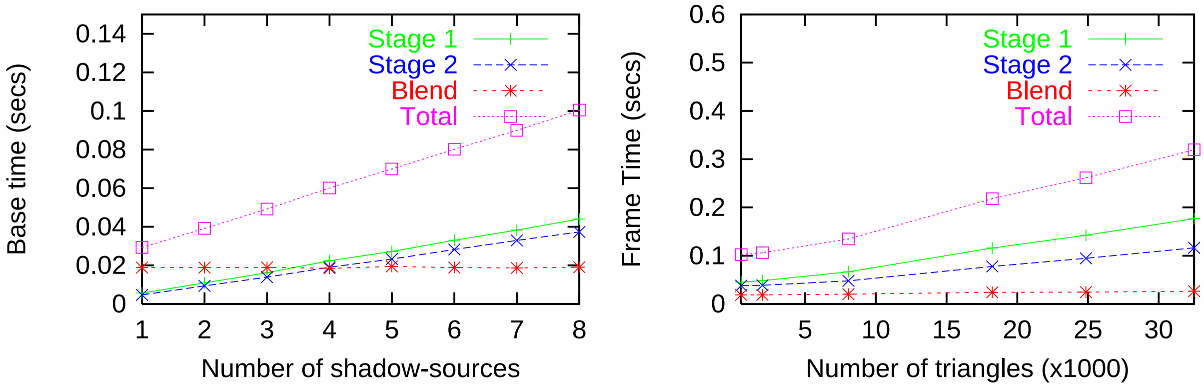

Figure 7: Timing breakdowns for a single frame.

almost 50,000 triangles, and was rendered at almost 2 fps with eight shadow-sources. The shadow intensities are generally a good match, but again, differences due to surface orientation and accessibility are visible.

Finally, Figure 7 gives timing breakdowns for a single frame. On the left we plot the total time required to generate a frame that contains no synthetic objects (and therefore represents the total cost of the image-processing operations we perform), for between 1 and 8 shadow-sources. This represents the minimum frame-time that we can achieve (over 16 fps with 4 shadow-sources, almost 10 fps with 8 sources). Also plotted are the total times for both stages of shadow generation, and the time required to generate and blend the background image. On the right, we show how frame-time increases for larger numbers of synthetic objects (this graph was produced using 8 shadow-sources). We have found that the time required to render a single frame increases linearly as the number of synthetic triangles increases. For 32,500 triangles, the frame-rate drops from almost 10 fps to just over 3 fps.

9 Conclusion

In this paper we have presented new algorithms that enable synthetic objects to be integrated into background images at interactive rates. We have combined recent advances in the image-based capture and storage of real-world illumination with novel shadow generation and image composition algorithms. Overall, we are able to manipulate complex synthetic objects at rates of several frames-per-second using a single graphics pipeline.

We have ignored inter-reflections between synthetic objects as well as reflections of the synthetic objects in the local scene. Also, our method is currently restricted to a static viewpoint. Camera movement could be allowed for using real-time tracking techniques, and by warping sphere-maps according to the changing camera orientation.

Acknowledgements

The authors would like to thank their colleagues in the Advanced Interfaces Group for helpful discussions and comments on this paper. The dragon model shown in Figure 6 was constructed by the Stanford University Computer Graphics Laboratory using a Cyberware 3030MS optical triangulation scanner, and is available from http://www-graphics.stanford.edu/data/3Dscanrep. This research was supported by EPSRC grant number GR/M14531, entitled 'REVEAL: Reconstruction of Virtual Environments with Accurate Lighting'.

References

[1] Azuma, R. T. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments 6*, 4 (August 1997), 355–385.

[2] Blinn, J. F., and Newell, M. E. Texture and reflection in computer generated images. *Communications of the ACM 19*, 10 (October 1976), 542—546.

[3] Boisvert, R. F., Howe, S. E., and Kahaner, D. K. NBS core math library (CMLIB), April 1998. Version 3.0, Center for Applied Mathematics, NBS, http://www.math.iastate.edu/docs/cmlib/cmlib.html.

[4] Cabral, B., Olano, M., and Nemec, P. Reflection space image based rendering. *Proceedings of SIGGRAPH 99* (August 1999), 165–170.

[5] Debevec, P. 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.

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

[7] Diefenbach, P. J., and Badler, N. I. Multi-pass pipeline rendering: Realism for dynamic environments. *1997 Symposium on Interactive 3D Graphics* (April 1997), 59–70.

[8] Drettakis, G., Robert, L., and Bougnoux, S. Interactive common illumination for computer augmented reality. *Eurographics Rendering Workshop 1997* (June 1997), 45–56.

[9] Fournier, A., Gunawan, A. S., and Romanzin, C. Common illumination between real and computer generated scenes. In *Proceedings of Graphics Interface '93* (San Francisco, CA, May 1993), Morgan Kaufmann, pp. 254–262.

[10] Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. Modelling the interaction of light between diffuse surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)* (July 1984), vol. 18, pp. 212–222.

[11] Greene, N. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications 6*, 11 (November 1986), 21–29.

[12] Haeberli, P., and Akeley, K. The accumulation buffer: Hardware support for high-quality rendering. *Computer Graphics (Proceedings of SIGGRAPH 90) 24*, 4 (August 1990), 309–318.

[13] Heidrich, W., and Seidel, H.-P. Realistic, hardware-accelerated shading and lighting. *Proceedings of SIGGRAPH 99* (August 1999), 171–178.

[14] Kajiya, J. T. The rendering equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)* (August 1986), vol. 20, pp. 143–150.

[15] Kempf, R., and Donnelly, J. H. OpenGL on Silicon Graphics systems, June 1998. http://toolbox.sgi.com/TaseOfDT/documents/OpenGL.

[16] Lewis, R. R. Making shaders more physically plausible. In *Fourth Eurographics Workshop on Rendering* (Paris, France, June 1993), no. Series EG 93 RW, pp. 47–62.

[17] Nakamae, E., Harada, K., Ishizaki, T., and Nishita, T. 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.

[18] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipies in C: The Art of Scientific Computing*, 2nd ed. ed. Cambridge University Press, Cambridge, NY, 1992.

[19] Reeves, W. T., Salesin, D. H., and Cook, R. L. Rendering antialiased shadows with depth maps. *Computer Graphics (Proceedings of SIGGRAPH 87) 21*, 4 (July 1987), 283–291.

[20] SATO, I., SATO, Y., AND IKEUCHI, K. 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.

[21] SEGAL, M., KOROBKIN, C., VAN WIDENFELT, R., FORAN, J., AND HAEBERLI, P. E. Fast shadows and lighting effects using texture mapping. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (July 1992), 249–252.

[22] STATE, A., HIROTA, G., CHEN, D. T., GARRETT, B., AND LIVINGSTON, M. Superior augmented reality registration by integrating landmark tracking and magnetic tracking. *Proceedings of SIGGRAPH 96* (August 1996), 429–438.

[23] THOMPSON, W. B., SHIRLEY, P., SMITS, B., KERSTEN, D. J., AND MADISON, C. Visual glue. Technical Report UUCS-98-007, University of Utah, March 1998.

[24] TSAI, R. Y. A versatile camera calibration technique for high accuracy machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation 3*, 4 (August 1987), 323–344.

[25] WALTER, B., ALPPAY, G., LAFORTUNE, E. P. F., FERNANDEZ, S., AND GREENBERG, D. P. Fitting virtual lights for non-diffuse walkthroughs. *Proceedings of SIGGRAPH 97* (August 1997), 45–48.

[26] WARD, G. J. The RADIANCE lighting simulation and rendering system. *Proceedings of SIGGRAPH 94* (July 1994), 459–472.