

Rapid Shadow Generation in Real-World Lighting Environments

Simon Gibson, Jon Cook, Toby Howard and Roger Hubbard[†]

Advanced Interfaces Group, University of Manchester, UK

Abstract

We propose a new algorithm that uses consumer-level graphics hardware to render shadows cast by synthetic objects and a real lighting environment. This has immediate benefit for interactive Augmented Reality applications, where synthetic objects must be accurately merged with real images. We show how soft shadows cast by direct and indirect illumination sources may be generated and composited into a background image at interactive rates. We describe how the sources of light (and hence shadow) affecting each point in an image can be efficiently encoded using a hierarchical shaft-based subdivision of line-space. This subdivision is then used to determine the sources of light that are occluded by synthetic objects, and we show how the contributions from these sources may be removed from a background image using facilities available on modern graphics hardware. A trade-off may be made at run-time between shadow accuracy and rendering cost, converging towards a result that is subjectively similar to that obtained using ray-tracing based differential rendering algorithms. Examples of the proposed technique are given for a variety of different lighting environments, and the visual fidelity of images generated by our algorithm is compared to both real photographs and synthetic images generated using non-real-time techniques.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image GenerationBitmap and framebuffer operations I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism-Color, shading, shadowing and texture.

1. Introduction

The ability to merge synthetically generated objects into photographs of a real scene is becoming central to many applications of computer graphics, and in particular, mixed or augmented reality. In many situations, this merging must be done at rates of many frames-per-second if an illusion of interactivity is to be maintained. Also, visually realistic combinations of objects and background images are required if the ultimate goal of augmentation is to present images to the user that are indistinguishable from reality. To achieve these goals the synthetic objects must be registered into the camera's coordinate frame, correctly composited with the image to resolve occlusions, and illuminated using the same lighting conditions as in the real scene. Finally, the augmentation process requires determining how the synthetic objects affect the illumination *already present* in the scene. Typically,

these changes in illumination take the form of reflections of the synthetic object in background surfaces, and occlusions of light transport paths that manifest themselves as shadows cast onto the real objects.

Traditionally, the competing requirements of real-time rendering and visual realism have meant that achieving interactive photorealistic Augmented Reality has been a distant goal. Recently however, techniques have been developed that allow synthetic objects to be illuminated with real-world lighting environments in real-time (see for example^{21, 26, 29}). Although these techniques are able to accurately shade surfaces, photorealism also requires that synthetic objects interact with the light affecting other *real* objects in the image. The addition of soft shadows in particular has been shown to increase the perception of image realism²⁴, and can also enhance users' spatial awareness^{32, 19}. In this paper we describe a new approach to solving the problem of interactive shadow generation for photorealistic Augmented Reality.

[†] e-mail: {sg|cookj|toby|roger}@cs.man.ac.uk



Figure 1: Interactive shading and shadowing of a 2,500 triangle synthetic object into a background photograph, running at over 11 frames-per-second on an NVIDIA GeForce4 GPU (left). For comparison, a ray-traced image is also shown, rendered in 1 hour using existing differential rendering algorithms (right).

Our algorithm uses geometry and illumination data captured using computer vision and high dynamic-range imaging techniques. We use a shaft-based data structure to provide a hierarchical subdivision of the light transport paths within the reconstructed environment. Shafts are used to link a hierarchy of source patches with a hierarchy of the receiver patches visible in the image, thereby allowing us to quickly determine the sources of light that are potentially occluded by any synthetic objects. We will show how hardware accelerated shadow-mapping may be used to identify the pixels in an image where light from these sources is occluded by synthetic objects. Multiple rendering passes are then performed that blend hard shadows together to approximate the soft shadow cast by the object. We will show how the contributions of light may be easily removed from the background image using facilities commonly found on modern graphics hardware. This results in a rendering algorithm capable of generating complex, visually realistic shadows at interactive frame-rates.

It is important to note the assumptions we are making in order to generate these shadows. Most significantly, we assume that a soft shadow can be accurately represented using multiple overlapping hard-edged shadows¹⁸. Whilst this is rarely true when using small numbers of hard shadows, we will show that our algorithm is capable of achieving interactive frame-rates whilst using a large number of shadow blending passes, which allows a much wider variety of soft shadows to be approximated. We also assume that the only moving objects are the synthetic ones we are introducing, and that casting shadows is the only effect these synthetic objects have on the environment.

Figure 1 shows an example of our algorithm in use. On the left is an image rendered at over 11 frames-per-second using an NVIDIA GeForce4 graphics card. By way of comparison, on the right is an image generated using ray-tracing and differential rendering techniques⁵. As the Figure shows, the shadows appear subjectively similar even though the ray-traced image took over 1 hour to generate.

The remainder of this paper is organised as follows: Section 2 describes related work, and Section 3 briefly outlines the techniques we use to capture real-world illumination. The main body of this paper contains a more detailed explanation of our new shadow rendering algorithm, presented in Sections 4 to 8. Results for a variety of lighting environments and shadow types are then given in Section 9, along with visual comparisons between our algorithm, ray-traced images and photographs. We also illustrate the graceful trade-off between image quality and rendering time our algorithm achieves. Finally, we draw conclusions and describe future work in Section 10.

2. Previous Work

There has been an enormous amount of research devoted to shadow generation. The literature is too large to review in this paper, but useful surveys can be found in³⁴ and¹⁶. Here, we will focus on previous work that is related to the problem of generating realistic shadows at interactive rates, or aims to composite shadows into a background photograph.

Basic shadow-mapping techniques³³ have been extended to generate soft shadows by approximating the penumbral regions using several hard-edged shadows³. By rendering each shadow from a slightly different position on the light source, and then combining the maps together, realistic representations of soft shadows can be generated. Alternative approaches that attempt to reduce the cost of soft shadow generation include convolution³⁰, “soft objects”²³ or search techniques² to approximate the penumbral region.

Radiosity^{4,28} has previously been used to generate soft shadows, but at a large computational cost. More recently, extensions to these techniques have been made to allow updates to localised regions of the solution, allowing for object movement (see, for example^{8,13,31}). Following pioneering work by Fournier *et al.*⁹, Drettakis *et al.*⁷ and Loscos *et al.*²² used an interactive cluster-based radiosity system to generate the shadows cast by a synthetic object in a real-environment, and composited those shadows into a background photograph at rates of 1 – 2 frames per second. Keller has also introduced the “Instant Radiosity” algorithm²⁰ that uses shadow-mapping hardware to accelerate the generation of globally illuminated environments.

The difficulty in applying hardware-based shadow-mapping to photorealistic Augmented Reality lies in the fact that real-world lighting environments contain a wide variety of different types of light sources, ranging from small focused spot-lights to broad area lights or even diffuse sky-light. As the number or area of light sources increases, it becomes harder to apply shadow-mapping and generate believable synthetic shadows. This is especially so if important secondary sources of illumination are required to cast shadows.

To deal with the problem of rendering with a wide vari-

ety of real-world light sources, Debevec proposed the use of *image-based lighting* techniques to allow real-world lighting environments to be captured and used to illuminate synthetic objects⁵. High dynamic-range images⁶ of a light probe were used in conjunction with a ray-tracing algorithm to render shadows cast by synthetic objects. Differential rendering techniques (discussed in more detail in Section 7) were used to produce photorealistic augmented images containing caustics and shadows. A similar algorithm was proposed by Sato *et al.*²⁷, with the light probe replaced by a camera with a hemispherical lens. Unfortunately, due to the compute intensive nature of the ray-tracing algorithms used in these approaches, interacting with the synthetic objects at rates required in Augmented Reality applications is not yet possible.

To achieve interactive update rates whilst rendering with real-world illumination, Gibson and Murta proposed using computer graphics hardware to render the shadows cast by synthetic objects¹². Shadows were approximated using multiple hard-edged shadow-maps, and blended into the background image using accumulation-buffer hardware. Although capable of generating images at rates of several frames-per-second, their approach assumed that all light sources in the scene were distant from the synthetic objects. Shadows cast by the objects were also only valid when falling onto a horizontal surface lying immediately below the object, limiting the applicability of the algorithm.

Unlike the techniques described in¹², the approach presented in this paper is not constrained by the assumption of distant light sources, allowing for more general lighting environments to be used. Shadows cast by the synthetic objects are also accurate for all orientations and positions of receiver surface. Finally, our algorithm is capable of trading accuracy against rendering time, enabling synthetic objects and subjectively realistic shadows to be merged into background images in real time.

3. Data Capture

Before we can describe our shadow generation algorithm, we will first outline the techniques we use to capture the illumination present in a real environment. Currently, we focus on augmenting real images captured using standard digital cameras. The examples shown in this paper are for single images, but the shadow rendering algorithm we propose is view-independent, and so could also be applied to moving cameras. Image-based modelling is used to reconstruct an approximate model of the background scene geometry¹¹ and high dynamic-range (HDR) photography⁶ is used to capture images of a lightprobe⁵. The transfer function of the camera is estimated, and HDR radiance information is mapped outwards from the lightprobe onto the approximate scene model. The scene geometry is then triangulated, and an approximate diffuse reflectance and radiance value calculated for each triangular patch. Note that both primary and non-

primary sources of light are accounted for in the triangulated scene model.

4. Overview

Shadow generation proceeds by first constructing a hierarchical subdivision of the line-space between all source and receiver patches in the scene. This hierarchy is then used to determine the sources of occluded radiance transfers between sources and receivers. Finally, shadows are generated by approximating the removal of each transfer from the background photograph.

Construction of the line-space subdivision relies on the patches in the environment being partitioned into two sets, containing source and receiver patches respectively. Note that a single patch may be classified as both a source and a receiver, and hence may appear in both sets. Also, we make no distinction between primary and non-primary sources of light, and simply take every patch with non-zero radiance as a potential member of the source set. In discussions below, we refer to any patch with a non-zero radiance as a “source patch”.

The *receiver set* contains all patches that are visible from the point of view of the calibrated image camera. The *source set* contains the patches that are considered to provide significant contributions of light to the image. This set is built by first sorting all patches in decreasing order of radiance. The source set is defined as the first N patches in the sorted list having a total power equal to a user-specified percentage of the total power of all patches. This has the effect of removing very insignificant sources of light from further consideration. The percentage of radiance can be used to trade accuracy against shaft hierarchy traversal time, but typically, a value of around 70% has been found to be satisfactory in all situations we have encountered, as this accounts for all primary and important secondary sources of light.

5. Radiance Transfer Pre-computation

One important assumption we make during the shadow rendering process is that the background environment remains static. This allows us to pre-compute the radiance transfer from each source patch to the vertices of patches contained in the receiver set. Assuming each source patch emits light diffusely, we calculate the form-factor between each source patch and each receiver vertex^{4,28}, multiplied by an estimate of the point-to-patch visibility obtained using ray-casting. Because an approximate reflectivity for the vertex has already been estimated, the radiance transfer from one source patch to each receiver vertex can be found, and stored with the source patch. These radiance transfers will be used during shaft-hierarchy traversal to identify shafts that represent insignificant transfers of light, and also during shadow compositing to remove the contributions of light emitted by occluded sources from the background image. Although this

is an $O(n^2)$ operation, radiance transfers can be calculated quickly in practice, due to the small number of receiver vertices and source patches.

6. Shaft-Hierarchy Construction

Before the shaft-hierarchy can be built, patches in the source and receiver sets must be clustered together into separate hierarchies. Patches in the receiver set are clustered using top-down octree subdivision. Subdivision is halted once a node contains less than a user-specified number of receiver patches. Typically, we build the hierarchy with at most 8 patches in one leaf node, but this number can be increased or decreased to trade accuracy against shaft-hierarchy traversal time. For the source set, it is important that we have fine-grain control over traversal of the source hierarchy (see Section 6.1 for further details). Because of this, we cluster patches in the source set using top-down binary KD-tree subdivision, which results in a much deeper hierarchy than with an octree. Subdivision is halted once a node contains a single source patch. For non-leaf nodes in the source hierarchy, the total radiance transfer from all child patches to each receiver vertex is calculated, summed, and stored with the node. This will be used in Section 8 when generating shadows from non-leaf positions in the hierarchy.

Once the source and receiver hierarchies are in place, the sets of line segments connecting nodes in the source and receiver hierarchies can be constructed using a hierarchy of shafts^{15, 8}. The purpose of the shaft hierarchy is to allow the sources of light that are potentially occluded by an object to be quickly identified.

Shaft-hierarchy construction proceeds in a relatively straightforward manner, starting with a shaft linking the root of the source hierarchy to the root of the receiver hierarchy. At each level the planes bounding the region of line-space between patches in the source and receiver nodes are stored with the shaft. Each shaft is recursively subdivided until the leaves of both the source and receiver hierarchies are reached. For each shaft, the total radiance transfer from its source patches to each of its receiver patch vertices is calculated. Recursion is terminated if it is found that the total radiance contribution from the shaft's source patches to each of its receiver vertices is less than 2% of the total radiance associated with the vertex. This avoids using many shafts to store visually insignificant contributions of light¹⁷, which in turn accelerates traversal of the shaft hierarchy and reduces memory requirements.

The shaft hierarchy introduced in this paper has certain similarities to that proposed by Drettakis and Sillion⁸. The main difference between the two approaches is that our hierarchy is only used to store a coarse representation of existing light transport paths in order to identify the source patches that are potentially affected by a moving object. Once these sets of patches have been identified, shadow mapping hardware is used to resolve the fine-grain occlusions of light (see

Section 7). Because we are encoding an existing static lighting solution, we are also able to remove shafts that transfer insignificant contributions of energy. This is in contrast to the hierarchy proposed by Drettakis and Sillion, which is used to encode the complete set of light transport paths in an environment. As will be demonstrated later, this separation of coarse and fine-level evaluation allows our shaft hierarchy to be constructed very quickly using a small amount of memory (see Section 9).

6.1. Hierarchy Traversal

In order to augment an image with shadows cast by a synthetic object, the sources of light occluded by the object must be rapidly identified. The shaft hierarchy described above is used to perform this task, and in this section we outline how a list of potentially occluded source patches may be generated.

Given the bounding box of a synthetic object at one particular instance in time, we are able to quickly identify the set of shafts that intersect this box and are therefore potentially occluded by the object. This traversal of line-space is done by visiting each node of the shaft-hierarchy recursively, starting at the root. An intersection test is applied between the shaft s and the object's bounding box¹⁵. If the box does not intersect s , further traversal of the portion of line-space associated with the shaft can cease. Alternatively, if an intersection occurs, the test is applied recursively to each of s 's children. If s is a leaf shaft then the source patch p associated with s is added to a list. p is then tagged with a frame-number counter that is incremented after every frame is rendered. As further source patches are found their counter tags are checked against the current frame number to make sure each patch is not added to the list multiple times. Once traversal of the shaft-hierarchy has been completed, we are left with a list of source patches that may cast shadows from the synthetic object (the *source list*). Similarly, by placing the receiver patches associated with the leaf shafts in a *receiver list*, we are also able to identify the regions of the scene that will potentially receive a shadow cast by the synthetic object.

The shadow compositing algorithm described in the next section generates a single hard shadow for each of these source patches, blending them together to form an approximation to the correct shadow. Typically, the time required to do this will exceed the amount of time the user is willing to spend generating each frame. For this reason, our rendering algorithm is able to use the source hierarchy to trade accuracy against frame-rate and generate single hard shadows from groups of source patches in order to render a single frame within the available time. The mechanisms by which this is achieved will be described in Section 8.

The memory requirements and time required to traverse line-space depend on the complexity of both the source and

receiver hierarchies. As mentioned above, we use an octree subdivision and large leaf size for the receiver hierarchy, and deeper KD-tree subdivision with a small leaf size (i.e. a single patch) for the source hierarchy. The octree subdivision of receivers results in a broad but relatively shallow receiver hierarchy, meaning that large regions of line-space may be quickly removed from consideration, and traversal to the leaf nodes occurs rapidly. For the source hierarchy, however, more fine-grain traversal is required in order to meet the required frame rate. Because subdivision of a binary KD-tree node only increases the total number of leaf nodes by one, this structure is used to store the source patch hierarchy.

7. Shadow Compositing

The process of compositing shadows into the background image occurs after the synthetic objects have been shaded and depth composited with the scene model. The overall approach we take is to generate a shadow-map for each patch in the source list, and use this shadow-map as a mask to remove the corresponding contribution of light from the background image in regions where the source is occluded from receivers by the synthetic object. This process is repeated for each source patch, blending multiple shadows into the background image and results in a subjectively realistic representation of the real shadow. By using facilities available on modern graphics hardware, the generation of these shadow-maps and the removal of light contributions from the background image can be done quickly enough to allow frames to be generated at interactive rates. In the following discussion we will assume that we are generating a single shadow from each patch in the source list. In Section 8 we will show how this assumption may be lifted, allowing the overall rendering speed and quality to be increased or decreased. The algorithm described here is a modification of the differential rendering algorithm introduced by Debevec⁵, enabling us to work with standard low dynamic-range frame buffers found in commonly available graphics hardware.

The differential rendering algorithm introduced by Debevec describes how two synthetic images of a scene may be used to compute the changes in a background photograph caused by the introduction of synthetic objects. Given a rendered image I_{obj} , containing the synthetic objects and scene geometry illuminated by the reconstructed lighting data, and a second image I_{noobj} that does not contain the synthetic objects, the difference between these two images, I_{ϵ} , is subtracted from the background photograph I_b :

$$I_{final} = I_b - I_{\epsilon} = I_b - (I_{noobj} - I_{obj}) \quad (1)$$

in order to generate a final image I_{final} that contains the correct shadowing effects. Wherever I_{obj} is darker than I_{noobj} (i.e. the areas where the synthetic object cast a shadow), light is subtracted from the background image accordingly.

More specifically, consider a pixel in the image, and a point x which corresponds to the nearest surface seen

```

1. Pre-process:
   For each source patch  $j$ 
     For each receiver vertex  $i$ 
       Calculate  $L_{ij}$ 

2. Repeat for each frame:
   Render the background image
   Render the synthetic objects
   For each source patch  $j$ 
     Enable shadow mapping to multiply by  $M_{ij}$ 
     Subtract contribution from  $j$  from the frame-buffer
     by rendering the receiver mesh with vertex colours
     set to  $L_{ij}$ 

```

Figure 2: Two stage compositing process for differential shadow rendering.

through that pixel. The adjustment ϵ_x that must be subtracted from the radiance associated with the pixel is simply:

$$\epsilon_x = \sum_{j=0}^{N-1} L_{xj} - \sum_{j=0}^{N-1} L_{xj} V_{xj} = \sum_{j=0}^{N-1} L_{xj} M_{xj} \quad (2)$$

where the summation is over all source patches $j = 0 \dots N - 1$, L_{xj} is the unoccluded radiance transferred from source j to x and then reflected at x towards the camera, and V_{xj} is the visibility of j with respect to x , i.e. $0 \leq V_{xj} \leq 1$, where $V_{xj} = 0$ if the transfer is completely occluded by a synthetic object, and 1 if it is completely visible. Defining a new term, $M_{xj} = 1 - V_{xj}$, allows the adjustment to be calculated using a single summation, where M_{xj} represents an *occlusion mask*, which varies between 1 when i is completely occluded from j , and 0 when it is completely visible.

In order to apply these adjustments to a background image, we assume that the background scene is static, and separate the term inside the summation in Equation 2 into two parts: L_{xj} which can be pre-computed for each x and j , and M_{xj} which depends on the position of the dynamic synthetic objects.

In order to execute this algorithm at rates fast enough for interactive applications, we take the basic approach of performing the image generation and subtraction operations in Equation 1 using graphics hardware. In the following discussion, we assume that the graphics hardware and frame-buffer are able to process HDR data. Once the basic algorithm is described, extensions that allow us to work with low dynamic-range (LDR) data will be presented in Section 7.1. Facilities to perform these LDR operations are available on NVIDIA GeForce3/4 graphics hardware, using extensions to OpenGL 1.2.

We first assume that the contribution of a single source patch j to each scene point x is smoothly varying, allowing us to store L_{ij} for each j at the vertices i of patches in

the receiver set. We let the graphics hardware linearly interpolate the values between each receiver vertex. Differential rendering of shadows into a background image can then be performed using the two-stage process presented in Figure 2. Note that we have explicitly separated the calculation of M_{ij} from the subtraction of L_{ij} . This is done because of the different rendering techniques are used to execute each loop: The first is evaluated using hardware shadow-mapping, approximating M_{ij} at each pixel in the image using binary visible/invisible values. Subtractive blending is then used during the second loop, and the receiver set is drawn with the colour of each vertex i set to L_{ij} . Texture combiners are set to use the shadow-map as a mask, simulating the multiplication by M_{ij} .

7.1. Shadow Compositing using Graphics Hardware

The discussion so far has only considered HDR representations of light where, assuming access to a floating-point frame buffer, we can operate entirely on floating-point radiance values and map back to pixel intensities as a post-process. Complications occur, however, when we try to apply differential rendering algorithms to LDR images, as used by most digital cameras and graphics hardware. Most importantly, for the background image we wish to augment, the relationship between high and low dynamic-range representations of light is non-linear. Ideally we would like to perform all operations using HDR data and apply a non-linear tone-map after shadow compositing:

$$I_{final} = T(L_{final}) = T(L_b - L_e)$$

where $I = T(L)$ is the tone-map transforming radiance into pixel colours. Unfortunately, due to the LDR nature of the frame-buffer we must operate entirely with LDR data.

By letting the graphics hardware interpolate between vertices in the receiver set, we can reduce the problem to one of performing differential rendering at the receiver vertices themselves. We will denote the desired HDR differential rendering process at a vertex i as:

$$I_{final_i} = T(L_i - \sum_{j=0}^{N-1} L_{ij}M_{ij})$$

where L_i represents the radiance obtained from the image at the pixel location associated with vertex i . Define a new *intensity transfer* S_{ij} for each pair of a vertex i and source patch j . These intensity transfers are LDR equivalents of the radiance transfers L_{ij} in Equation 2. We wish to subtract these intensity transfers from the LDR frame-buffer intensity I_i so that the overall result is equivalent to when HDR operations are used:

$$I_{final_i} = I_i - \sum_{j=0}^{N-1} S_{ij}M_{ij} \quad (3)$$

Because we will be removing these contributions from the

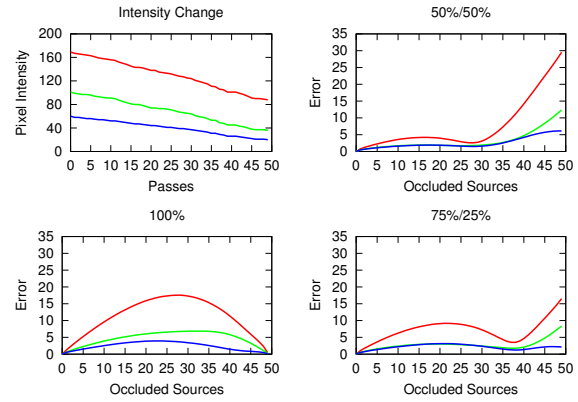


Figure 3: The reduction in frame-buffer intensity as increasing number of shadow passes are applied (top-left), and the error (in pixel colour) caused by the assumption that all patches are occluded in two equally-sized sets (top-right). Errors for two sets of different sizes are shown below (bottom row).

frame-buffer using multiple rendering passes, and we do not know the correct values for M_{ij} . Equation 3 implies that:

$$S_{ik} = I_i - T(L_i - \sum_{j=0}^k L_{ij}M_{ij}) - \sum_{j=0}^{k-1} S_{ij}M_{ij} \quad (4)$$

must hold for each $0 \leq k < N$. Unfortunately, we are unable to pre-compute the intensity transfers exactly from this relation, because the values of M_{ij} are not known until rendering occurs. The non-linearity of $T()$ also means that the final result is dependent on the order the source patches are considered. We can, however, generate a useful approximation by assuming that each source patch is either entirely visible or entirely invisible. Initially, we don't know which of the source patches will be visible and which will be invisible, but if we assign estimates to each source patch then we can calculate S_{ij} and remove the correct contribution from the background image. If the visibility estimates were correct, this should result in a correct final image, assuming the order that the source patches are considered remains the same. In practice, the order is unlikely to remain fixed, but if we choose to order patches from brightest to dimmest when evaluating Equation 3, and ensure we sort any later sets of source patches in that same order, the approximation error will be reduced.

Without knowing which patches are actually occluded, we can generate an approximation by randomly partitioning the source patches into two separate sets. By assuming that when all the patches in the first set are occluded those in the second remain visible, we can fix the values of M_{ij} and calculate intensity transfers for the first set of patches using Equation 4. Similarly, assuming that when the patches in the second set are all occluded, those in the first set are visible, we can determine the remaining intensity transfers.

Figure 3 illustrates how this approximation affects the final shadow intensity for differently sized sets. The graph in the top-left shows the typical reduction in I_i that occurs after each successive rendering pass using a set of 50 random source patches. The remaining graphs plot the error found when assuming that all source patches are occluded in differently sized sets. Intensity transfers were calculated as described above. Varying numbers of P ($0 \leq P \leq 50$) source patches were then randomly selected as being *actually occluded*, simulating the evaluation of M_{ij} using shadow-mapping (plotted on the horizontal axis of each graph). For each P , 10000 trials were run over 4 datasets, and P random patches were selected for each trial. The difference between the left and right-hand sides of Equation 3 was then measured, with $M_{.j} = 1$ for the P random patches, and 0 otherwise. The graph shows the variance of the error in red green and blue pixel intensities.

For each set size, the error is insignificant for small P . This is because subtracting a small number of incorrect intensity transfers has little effect on the overall image. Similarly, the error is also small for values of P that match the assumption being made (e.g. the error is small for $P = 25$ when assuming an 50%/50% split). For intermediate values of P , the error rises as increasing numbers of incorrect intensity transfers are subtracted from the image.

In practice, we have found that for receivers in the vicinity of synthetic objects, typical occlusion rates run at around 30 – 50% for the scenes we have examined, and only rarely rise above 75%. For this reason we have used the 50%/50% split in all further examples because this split has the smallest overall error in the 30 – 50% region (see the top-right graph in Figure 3).

7.1.1. Calculating Intensity Transfers

Intensity transfers can be calculated very quickly for each frame before the shadows are composited into the background image. Before these intensity transfers can be determined, the patches in the source list for the current frame are sorted in decreasing order of average radiance transfer to patches in the receiver set. The average transfer of radiance from each source patch can easily be pre-computed and stored with the source hierarchy because we assume that light reflected off the synthetic objects does not affect the overall illumination in the scene. The transfers can then be calculated using the algorithm presented in Figure 4. For each receiver vertex, V_1 and V_2 are initialised to the total radiance gathered from all source patches and reflected at the vertex towards the camera. These two radiance values will be used to calculate the intensity transfers under the assumption that the source patches are occluded in two equally sized sets, as described above. These initial radiance values are mapped to pixel colours C_1 and C_2 using the calibrated camera response function $T()$.

A loop is then made over all patches in the source list that

```

1. Pre-process:
   Sort source patches in decreasing order of transfer

2. For each receiver vertex  $i$ :
    $V_1 = V_2 = L_i$ 
    $C_1 = C_2 = T(L_i)$ 
   For each contributing source patch  $j$ :
     if  $j$  is even
        $V_1 = V_1 - L_{ij}$ 
        $C'_1 = T(V_1)$ 
        $S_{ij} = C_1 - C'_1$ 
        $C_1 = C'_1$ 
     else
        $V_2 = V_2 - L_{ij}$ 
        $C'_2 = T(V_2)$ 
        $S_{ij} = C_2 - C'_2$ 
        $C_2 = C'_2$ 
   endif

```

Figure 4: Pseudo-code for estimating intensity transfers, executed before drawing each frame.

can contribute radiance to the vertex. In order to quickly simulate a random assignment of patches to sets, we assign each patch according to a randomly generated id number between 0 and $N - 1$. For even numbered ids, the pre-calculated radiance transfer from the source to the receiver is subtracted from V_1 , and the radiance is then transformed by $T()$ into a pixel colour C'_1 . The intensity transfer S_{ij} is then calculated as the difference between C_1 and C'_1 . C_1 is set equal to C'_1 and the process repeated for the next source patch. For odd numbered ids the calculations are performed using V_2 and C_2 , so as the source list is traversed, two independent radiance values are used to estimate the intensity transfers. Each of these independent values corresponds to one of the sets we made in the occlusion assumption described above.

7.2. Shadow-Map Generation

As a pre-process, simplified representations of all synthetic objects are generated using the techniques described in¹⁰, each containing between 100 and 500 triangles. These simplified objects are used during shadow-map rendering, and shadow-map resolution is also limited to 256x256 pixels. This greatly accelerates rendering speed without visibly reducing image quality.

Once the intensity transfers have been estimated for the current frame, the second inner-loop of the algorithm presented in Figure 2 can be executed, with L_{ij} replaced by the transfers S_{ij} . The receiver set is drawn with vertex colours set to S_{ij} , and graphics hardware used to interpolate between these values. A shadow map is then generated for each source j , allowing us to find $M_{.j}$. This is done by first initialising the OpenGL projection and model-view matrices so

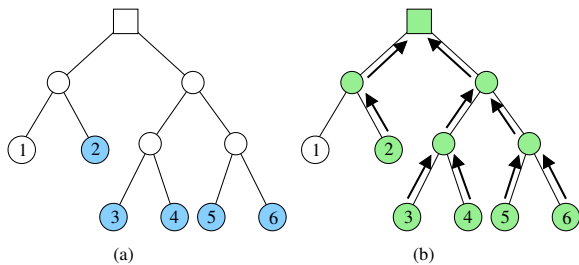


Figure 5: (a) Traversal of the shaft-hierarchy identifies 5 out of 6 potentially occluded source patches (marked in blue). (b) The affected portions of the source hierarchy (shown in green) are then identified by pushing a frame identification tag up towards the root node.

the synthetic object is contained entirely within the shadow-map, as seen from the source patch. The simplified representation of the synthetic object is then rendered into the depth buffer to produce the shadow-map. Hardware shadow-mapping, texture combiners, and blending operations are initialised so that when the geometric representation of the receiver set is drawn, the vertex colours (S_{ij}) are multiplied by M_j , and the product is subtracted from the background colour buffer. If required, self-shadows cast onto the synthetic objects can also be generated by approximating the intensity transfer S_{ij} from a source patch to the vertices of the object, and then rendering the object with shadow-mapping and blending enabled.

8. Controlling Frame-Rate

In the previous section we described how a shadow from each source patch could be generated and composited into a background photograph using commonly available graphics hardware. In interactive settings, the time required to do this for all source patches will often exceed the time a user is willing to spend generating a single frame. In these situations, what is required is a trade-off between overall shadow quality and rendering cost, and this can be achieved by generating shadow-maps from non-leaf nodes in the source hierarchy that was described in Section 6.

Figure 5(a) shows a typical source hierarchy, with a root node at the top and 6 source patches at the leaves. Assume, for example, that during construction of the source set, $N = 5$ out of the 6 source patches (shown in blue) have been identified as being potentially occluded by a synthetic object. Assuming that we are unable to generate shadows from all 5 sources due to frame-rate constraints, we need to find a *representative source set* that encapsulates the effect of all source patches and yet can be processed in the available time. We can do this very quickly before each frame is rendered using the algorithm described in this section.

Building the representative source set starts by pushing

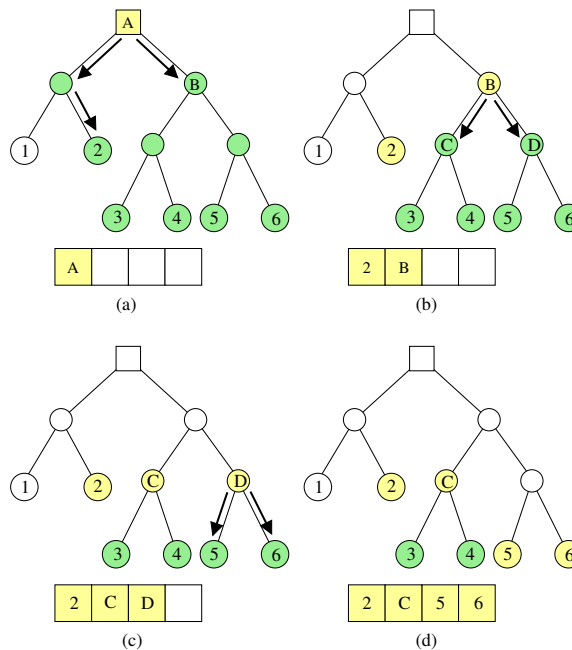


Figure 6: A set of $n = 4$ source clusters are identified that represent the combined effect of all 6 potentially occluded source patches. Starting at the root node (a), the affected portion of the hierarchy is traversed (b), (c), and (d) until the required number of source clusters are identified (yellow).

the current frame number from each potentially occluded source patch up towards the root of the hierarchy. This allows the branches of the hierarchy containing these patches to be identified and marked (shown in green in Figure 5(b)). Starting with the root, we wish to build a list of $n < N$ nodes, where each node can be either a leaf of the source hierarchy or an intermediate node representing the combined effect of several leaves.

Figure 6(a) shows the start of the construction process for $n = 4$ nodes. While the target number of nodes has not been reached, the node in the list which transfers the largest average amount of radiance to patches in the receiver set is removed from the list. This is done very quickly by storing the list using a binary tree, sorted by the average radiance transfer. Initially, as it is the only node in the list, the root node is removed. The hierarchy is then traversed by one level and the node's immediate children in the marked portions of the hierarchy are identified and added to the list (source patch "2" and node "B"). This process is repeated until the required number of patches or nodes is found. Figures 6(b) and (c) show further traversals, until finally, in (d) we reach the target of $n = 4$ nodes. Note that although we have chosen fewer than 5 nodes, the energy from all potentially occluded source patches is still accounted for, because the radiance transfer from node "C" represents the combined

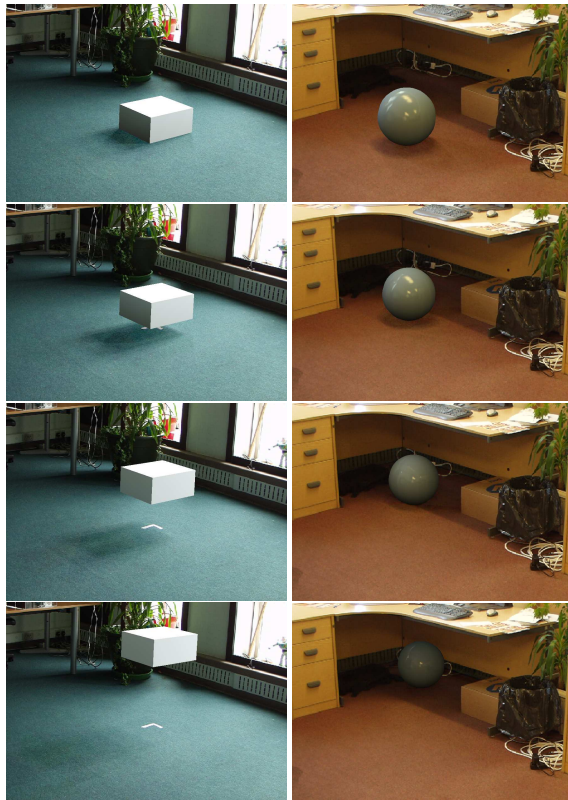


Figure 7: Examples of interaction between a synthetic object and real environment, generated at around 15 frames-per-second using our system. The left-hand column illustrates the reduction in shadow intensity that occurs as the synthetic object is raised off the ground. The sequence in the right-hand column shows how real and virtual shadows can interact as the synthetic object is moved underneath the real desk.

effect of source patches “3” and “4”. When rendering a single shadow-map from a cluster of sources, such as node “C”, the origin of the source is chosen to coincide with the centre of the patch that contributes the most energy to the receivers.

9. Results

The algorithm described in this paper has been implemented using OpenGL on a 2.5 GHz Pentium 4 PC running Microsoft Windows XP and equipped with a NVIDIA GeForce 4 Ti4600 graphics card. For all examples shown, synthetic objects have been shaded using a combination of an irradiance volume¹⁴ and dynamic environment-maps for specular reflections. The irradiance-volume uses spherical harmonics to represent irradiance²⁵, greatly reducing the computation and storage requirements.

Examples showing interactive object movement are given in Figure 7. All images are snapshots from an interactive session rendered at approximately 15 frames-per-second, using

50 blending passes. Of this, the majority of the time was spent generating and blending shadows into the background image, and the time required to shade each object was negligible. The left-hand column shows an example where the user is lifting a box off the floor of the scene. Because the intensity of the shadows blended into the background image are based on the actual amount of light transferred from source to receiver, the reduction in intensity of the synthetic shadow is correctly modelled as the object is raised off the floor. The right-hand column shows a different kind of interaction between a synthetic object and the environment, where the sphere moves under a real table. Notice the darkening of the object and merging of the synthetic shadow with the real shadow due to the fact that the desk has been included in the geometric scene model. Further examples are given in the accompanying video material.

Figure 8 compares rendering quality against ray-traced and photographic references for different lighting environments. In each row, an image produced using our interactive system is shown on the left, a ray-traced image generated using a HDR differential rendering algorithm⁵ is shown in the middle, and a photograph of a real object in the scene is shown on the right. The left-hand images were all generated using 50 blending passes, and were rendered at 14, 12 and 16 frames-per-second respectively. For comparison, the ray-traced images in the middle column each took several hours to generate using an un-optimized Monte-Carlo ray-tracer. Overall, the shadows generated using our algorithm are subjectively very similar to both the ray-traced and photographic references.

Note that the differences in shading of the synthetic objects in these examples are due to the fact that we did not accurately measure or model the reflectance properties of the real object. As such, the shading is only an approximation and these images are only intended to indicate the quality of the shadows that our rendering algorithm can generate.

The time spent constructing the patch and shaft hierarchies for these examples was relatively small. The first example in Figure 8 required around 30 seconds of pre-processing time, and produced a shaft hierarchy with 47,000 leaf nodes occupying just over 9 Mb of memory. The second and third examples only required 15 seconds of pre-processing, generating 102,000 and 64,000 leaf shafts respectively. Typically, each scene contained between 1000 and 2000 patches, and between 10 and 40 milliseconds was required to traverse the shaft hierarchy and generate a representative source set for each frame.

The trade-off that can be made between frame rendering time and shadow accuracy is illustrated under two different lighting environments in Figure 9. On the top row, a 500 triangle sphere was rendered into a background environment using different numbers of blending passes. For each number of passes, the algorithm described in Section 8 was used to determine a representative set of source nodes, and a sin-

gle shadow-map was generated for each node. From left to right, the Figure shows frames generated with 10, 20 and 50 passes. These were rendered at rates of 35, 25 and 14 frames-per-second. For comparison, a ray-traced image was also produced in approximately 1 hour and is shown on the right. Note that because the image on the far-left was generated with a smaller number of blending passes, the hard-edged shadows are clearly visible. Our algorithm is, however, able to maintain the same overall intensity of the shadow as in the ray-traced image (see Section 7.1). As the number of blending passes increases, the hard edges of the individual shadow-maps are less visible and the overall result becomes an increasingly better approximation to the ray-traced reference image on the far-right. Similar images for a second lighting environment are given on the bottom row. Frame rendering rates for this example were 27, 18 and 9 frames-per-second respectively, with the ray-traced image requiring over 1.5 hours to render.

10. Conclusions and Future Work

In this paper we have presented a new shadow rendering algorithm suitable for photorealistic Augmented Reality, where shadows cast by synthetic objects are composited into a background image fast enough to allow interactive object manipulation. Our algorithm combines a shaft-based hierarchical data structure that allows the rapid identification of the sources of light that are occluded by a synthetic object, with a technique that allows soft shadows to be approximated using multiple shadow-maps and blended into the background image using commonly available computer graphics hardware.

We have shown that we can generate subjectively realistic augmented images at interactive rates for a variety of different real-world lighting environments including both interior and natural illumination. Our algorithm is also capable of trading image accuracy against frame-rate by approximating shadows using different numbers of shadow-maps. As the number of shadow blending passes (and hence frame generation time) increases, the result rapidly approaches the quality obtained using a non-real-time differential rendering algorithm. Future work includes a more formal evaluation of the perceptual fidelity of the images, when compared to both photographs and real environments.

There are currently limitations in our system on the types of light sources that can be modelled. For example, we are unable to render shadows cast by direct sunlight, or other types of directional illumination. There is nothing inherent in the rendering algorithm preventing this, but our current methods of data capture (Section 3) are not able to distinguish between directional and diffuse sources of light in the scene. We also assume that all surfaces onto which shadows are cast are diffuse, although this is not a fundamental limitation of the algorithm. Because we pre-compute the radiance reduction caused by the occlusion of each source

of light (Section 5), a view-dependent evaluation of this could account for non-diffuse reflectance properties. However, such extensions are left as future work, mainly because of the complexity of recovering non-diffuse surface reflectance data for real-world environments^{35, 1}.

Although we have presented examples showing augmentation of static images, our shadow generation algorithm is not view-dependent in any way, and the techniques presented in this paper could also be applied to moving cameras. Finally, the overall rendering quality will be enhanced by the appearance of floating-point graphics pipelines in the next generation of computer graphics hardware. This will reduce rounding errors that can sometimes occur when blending large numbers of very faint shadows into the background image.

Acknowledgements

We would like to acknowledge the European Union for funding this work, as part of the ARIS project (IST-2000-28707), which is examining the application of Augmented Reality to interior design. We are also grateful to the other ARIS project partners for their support and assistance (Fraunhofer IGD, Intracom, INRIA-Loria, University of Bristol, and Athens Technology Center). The bunny model used in Figure 7 is available from the Stanford 3D Scanning Repository. Finally, we would like to thank the anonymous reviewers of a previous version of this paper for their helpful comments and suggestions.

References

1. Samuel Boivin and André Gagalowicz. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 107–116, August 2001. 10
2. Stefan Brabec and Hans-Peter Seidel. Single sample soft shadows using depth maps. In *Proc. Graphics Interface*, pages 219–228, May 2002. 2
3. Lynne S. Brotman and Norman I. Badler. Generating soft shadows with a depth buffer algorithm. *IEEE Computer Graphics & Applications*, 4(10):71–81, October 1984. 2
4. Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993. 2, 3
5. Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 189–198, Orlando, Florida, July 1998. 2, 3, 5, 9
6. Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual

- Conference Series, pages 369–378, Los Angeles, California, August 1997. 3
7. George Drettakis, Luc Robert, and Sylvain Bougnoux. Interactive common illumination for computer augmented reality. In *Proc. Eurographics Rendering Workshop 1997*, pages 45–56, St. Etienne, France, June 1997. 2
 8. George Drettakis and François X. Sillion. Interactive update of global illumination using a line-space hierarchy. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 57–64, Los Angeles, California, August 1997. 2, 4
 9. Alain Fournier, Atjeng S. Gunawan, and Chris Romanzin. Common illumination between real and computer generated scenes. In *Graphics Interface '93*, pages 254–262, May 1993. 2
 10. Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 209–216, August 1997. 7
 11. Simon Gibson, Roger J. Hubbard, Jon Cook, and Toby L. J. Howard. Interactive reconstruction of virtual environments from video sequences. *Computers & Graphics*, 27(3), April 2003. 3
 12. Simon Gibson and Alan Murta. Interactive rendering with real world illumination. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 365–376, June 2000. 3
 13. Xavier Granier and George Drettakis. Incremental updates for rapid glossy global illumination. *Computer Graphics Forum*, 20(3):268–277, 2001. 2
 14. Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics & Applications*, 18(2):32–43, March-April 1998. 9
 15. Eric Haines. A shaft culling tool. *Journal of Graphics Tools*, 5(1):23–26, 2000. 4
 16. Eric Haines and Tomas Möller. Real-time shadows. In *Game Developers Conference*, March 2001. 2
 17. Selig Hecht. The visual discrimination of intensity and the weber-fechner law. *Journal of General Physiology*, 7, 1924. 4
 18. Paul S. Heckbert and Michael Herf. Simulating soft shadows with graphics hardware. Technical Report CMU-CS-97-104, CS Department, Carnegie Mellon University, January 1997. 2
 19. Helen H. Hu, Amy A. Gooch, William B. Thompson, Brian E. Smits, John J. Rieser, and Peter Shirley. Visual cues for imminent object contact in realistic virtual environments. In *IEEE Visualization 2000*, pages 179–185, October 2000. 1
 20. Alexander Keller. Instant radiosity. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 49–56, 1997. 2
 21. Lutz Latta and Andreas Kolb. Homomorphic factorization of brdf-based lighting computation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):509–516, July 2002. 1
 22. Céline Loscos, Marie-Claude Frasson, George Drettakis, Bruce Walter, Xavier Granier, and Pierre Poulin. Interactive virtual relighting and remodeling of real scenes. In *Proc. Eurographics Rendering Workshop 1999*, Granada, Spain, June 1999. 2
 23. Steven Parker, Peter Shirley, and Brian Smits. Single sample soft shadows. Technical Report UUCS-98-019, Computer Science Department, University of Utah, October 1998. 2
 24. Paul Rademacher, Jed Lengyel, Ed Cutrell, and Turner Whitted. Measuring the perception of visual realism in images. In *Proc. 12th Eurographics Workshop on Rendering*, pages 235–248, Eurographics, June 2001. 1
 25. Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 497–500, August 2001. 9
 26. Ravi Ramamoorthi and Pat Hanrahan. Frequency space environment map rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):517–526, July 2002. 1
 27. Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Acquiring a radiance distribution to superimpose virtual objects onto a real scene. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):1–12, January - March 1999. 3
 28. Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, 1994. 2, 3
 29. Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):527–536, July 2002. 1
 30. Cyril Soler and François X. Sillion. Fast calculation of soft shadow textures using convolution. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 321–332, July 1998. 2
 31. Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2002)*, 21(3):537–546, July 2002. 2
 32. Leonard R. Wagner, James A. Ferwerda, and Donald P. Greenberg. Perceiving spatial relationships in computer generated images. In *IEEE Computer Graphics and Applications*, volume 21, pages 30–50, May 1992. 1
 33. Lance Williams. Casting curved shadows on curved surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)*, volume 12, pages 270–274, August 1978. 2
 34. Andrew Woo, Pierre Poulin, and Alain Fournier. A survey of shadow algorithms. *IEEE Computer Graphics and Applications*, 10(6):13–32, November 1990. 2
 35. Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 215–224, Los Angeles, California, August 1999. 10

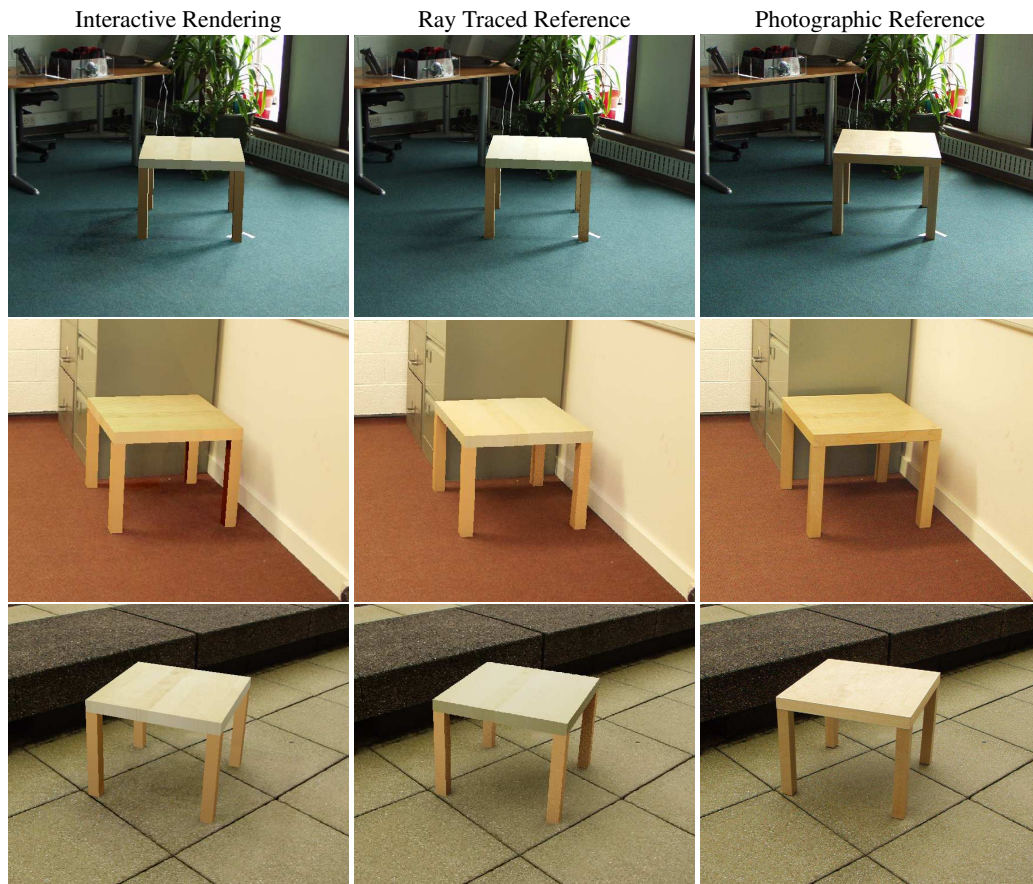


Figure 8: A comparison of image quality for three different scenes, containing both soft and harder-edged shadows cast by daylight and artificial light sources. Snapshots from interactive sessions with our system are shown on the left, generated at (from top to bottom) 14, 12 and 16 frames-per-second respectively. Ray-traced reference images are shown in the middle column, and photographic references containing an equivalent real object at approximately the same position are shown on the right. Further details are given in the text.

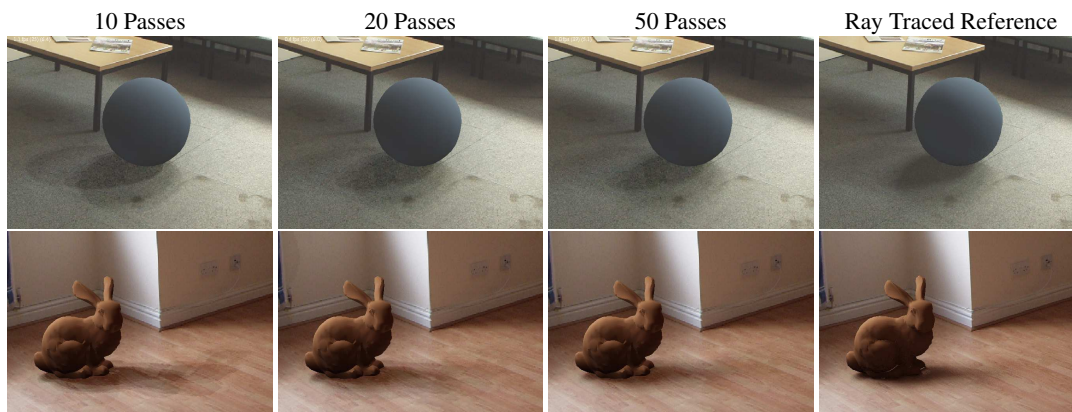


Figure 9: This figure shows the trade-off between rendering speed and accuracy that is available with our system, for two different lighting environments. From left to right, this figure presents snapshots from our system with shadows generated using 10, 20 and 50 blending passes. These images were rendered at approximately 35, 25 and 14 frames per second for the top row, and 27, 18 and 9 frames per second for the bottom row. For comparison, ray-traced reference images are shown on the far-right.