Image-based Lighting

09

9.1 INTRODUCTION

The previous chapters in this book have described numerous properties and advantages of HDR imagery. A major advantage is that HDR pixel values can cover the full range of light in a scene and can be stored as calibrated linear-response measurements of incident illumination. Earlier chapters have described how these images are useful for improved image processing, and for determining how a human observer might perceive a real-world scene, even if shown on an LDR display.

This chapter describes how HDR images can be used as sources of illumination for computer-generated objects and scenes. Because HDR images record the full range of light arriving at a point in space, they contain information about the shape, color, and intensity of direct light sources, as well as the color and distribution of the indirect light from surfaces in the rest of the scene. Using suitable rendering algorithms, we can use HDR images to accurately simulate how objects and environments would look if they were illuminated by light from the real world. This process of using images as light sources is called *image-based lighting* (IBL). In that IBL generally involves the use of HDR images, both the IBL process and the HDR images used for IBL are sometimes referred to as HDRI for high-dynamic-range imagery.

Figure 9.1 compares a simple scene illuminated by a traditional computer graphics light source (a) to its appearance as illuminated by three different image-based lighting environments (b through d). The scene's geometry consists of simple shapes and materials such as plastic, metal, and glass. In all of these images, the

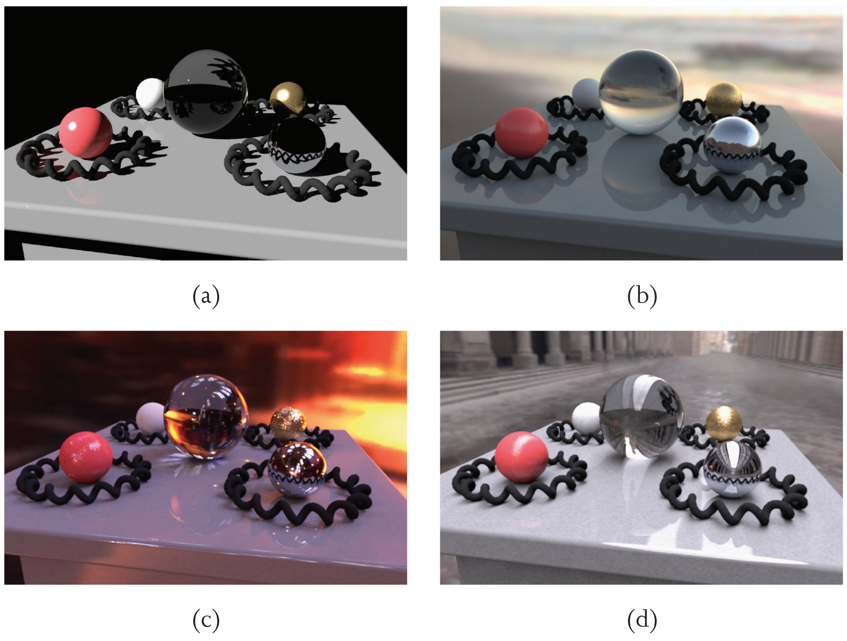


FIGURE 9.1 *Scene illuminated with (a) a traditional point light source. Scene illuminated with (b–d) HDR image-based lighting environments, including (b) sunset on a beach, (c) inside a cathedral with stained-glass windows, and (d) outside on a cloudy day.*

lighting is being simulated using the RADIANCE global illumination system [205]. Without IBL (a), the illumination is harsh and simplistic, and the scene appears noticeably computer generated. With IBL (b through d), the scene's level of realism and visual interest are increased — the shadows, reflections, and shading all exhibit complexities and subtleties that are realistic and internally consistent. In each rendering, a view of the captured environment appears in the background behind the objects. As another benefit of IBL, the objects appear to actually belong within the scenes that are lighting them.

In addition to HDR photography, IBL leverages two other important processes. One of them is omnidirectional photography, the process of capturing images that see in all directions from a particular point in space. HDR images used for IBL generally need to be omnidirectional, because light coming from every direction typically contributes to the appearance of real-world objects. This chapter describes some common methods of acquiring omnidirectional HDR *images*, known as *light probe images* or *HDR environment maps*, which can be used as HDR image-based lighting environments.

The other key technology for IBL is *global illumination*: rendering algorithms that simulate how light travels from light sources, reflects between surfaces, and produces the appearance of the computer-generated objects in renderings. Global illumination algorithms simulate the interreflection of light between diffuse surfaces, known as *radiosity* [170], and can more generally be built on the machinery of *ray tracing* [206] to simulate light transport within general scenes according to the *rendering equation* [175]. This chapter describes how such algorithms operate and demonstrates how they can be used to illuminate computer-generated scenes and objects with light captured in light probe images.

An important application of IBL is in the area of motion picture visual effects, where a common effect is to add computer-generated objects, creatures, and actors into filmed imagery as if they were really there when the scene was photographed. A key part of this problem is to match the light on the CG elements to be plausibly consistent with the light present within the environment. With IBL techniques, the real illumination can be captured at the location the CG object needs to be placed, and then used to light the CG element so that it has the same shading, shadows, and highlights as if it were really in the scene. Using this lighting as a starting point, visual effects artists can augment and sculpt the image-based lighting to achieve effects that are both dramatic and realistic.

Within the scope of IBL, there are several variants and extensions that increase the range of application of the technique. When implemented naïvely in global illumination software, IBL renderings can be computationally expensive for scenes with concentrated light sources. This chapter presents both user-guided and automatic techniques for *importance sampling* that make IBL calculations more efficient. In addition, various approximations of IBL can produce convincing results for many materials and environments, especially under appropriate artistic guidance. One is *environment mapping*, a precursor to IBL that yields extremely efficient and often convincing renderings by directly mapping images of an environment onto object surfaces. Another technique, *ambient occlusion*, uses some of IBL's machinery to approximate an object's self-shadowing so that it can be quickly applied to different lighting environments. To begin, we will start with a detailed example of IBL in a relatively basic form.

9.2 BASIC IMAGE-BASED LIGHTING

This section describes image-based lighting in both theoretical and practical terms using the example of *Rendering with Natural Light* (RNL), an IBL animation shown at the SIGGRAPH 98 Electronic Theater. The RNL scene is a still life of computer-generated spheres on a pedestal, and is illuminated by light captured in the Eucalyptus grove at the University of California at Berkeley. The animation was modeled, rendered, and illuminated using the RADIANCE lighting simulation system [205], and the necessary scene files and images for creating the animation are included on the companion DVD-ROM. The animation was created via the following steps:

1. Acquire and assemble the light probe image.
2. Model the geometry and reflectance of the scene.
3. Map the light probe to an emissive surface surrounding the scene.
4. Render the scene as illuminated by the IBL environment.
5. Postprocess and tone map the renderings.

9.2.1 ACQUIRE AND ASSEMBLE THE LIGHT PROBE

The lighting environment for RNL was acquired in the late afternoon using a three-inch chrome bearing and a digital camera. The mirrored ball and a digital video camera were placed on tripods about 4 feet from each other and 3.5 feet off the ground.[1] The digital video camera was zoomed until the sphere filled the frame, and the focus was set so that the reflected image was sharp. The camera's aperture was narrowed to f/8 to allow for sufficient depth of field, and an HDR image series was acquired with shutter speeds varying from $\frac{1}{4}$ second to $\frac{1}{10{,}000}$ second, spaced one stop apart. To cover the scene with better sampling, a second series was acquired after having moved the camera 90 degrees around to see the ball from the side (see Section 9.3.1). The process took only a few minutes and the resulting image series can be seen in Figures 9.2(a) and 9.2(c).

Mirrored spheres reflect nearly the entire environment they are in—not, as sometimes assumed, just the hemisphere looking back in the direction of the camera. This follows from the basic mirror formula that the angle of incidence is equal to the angle of reflection: rays near the outer edge of a sphere's image have an angle of reflection toward the camera that nears 90 degrees, and thus their angle of incidence also nears 90 degrees. Thus, the ray's angle of incidence relative to the camera nears 180 degrees, meaning that the rays originate from nearly the opposite side of the sphere relative to the camera.

Each image series of the sphere was converted into an HDR image using the HDR image assembly algorithm in Debevec and Malik [164] (Chapter 4), and the images were saved in RADIANCE's native HDR image format (Chapter 3). The algorithm derived the response curve of the video camera and produced HDR images where the pixel values were proportional to the light values reflected by the mirrored sphere. The total dynamic range of the scene was approximately 5,000:1, measuring from the dark shadows beneath the bushes to the bright blue of the sky and the thin white clouds lit from behind by the sun. As another measure of the range, the brightest pixel values in the sky and cloud regions were some 150 times the average level of light in the scene. The two views of the sphere were combined (using techniques presented in Section 9.3.1) and mapped into the angular map space (described in Section 9.4.2) to become the light probe image seen in Figure 9.2(b).

1 Many tripods allow the center pole (the vertical pole the tripod head is attached to) to be removed from the legs and reinserted upside-down, leaving the end of the pole pointing up and able to accommodate a mirrored sphere.

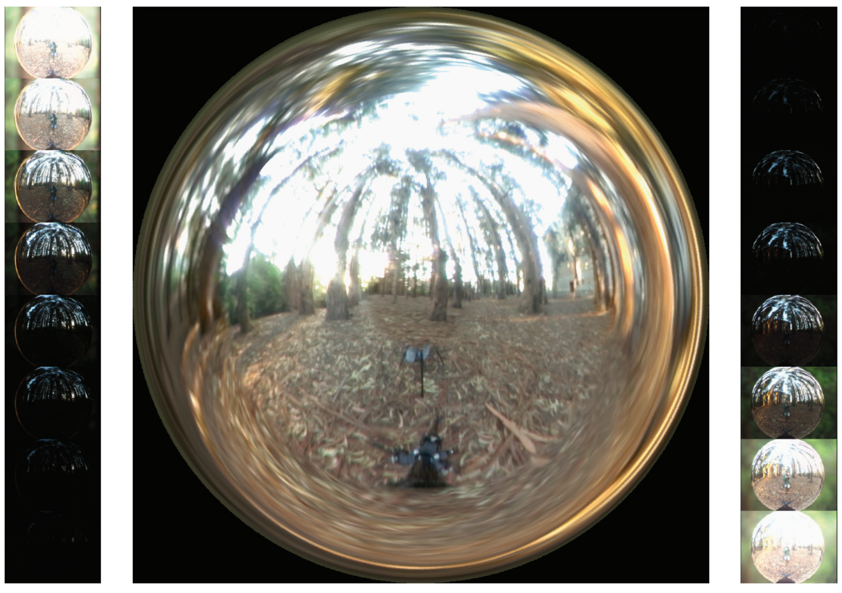


FIGURE 9.2 *The two HDRI series (a and c) used to capture the illumination in the Eucalyptus grove for RNL, and the resulting combined light probe image (b), converted into the Angular Map format.*

9.2.2 MODEL THE GEOMETRY AND REFLECTANCE OF THE SCENE

The RNL scene's spheres, stands, and pedestal were modeled using RADIANCE's standard scene primitives and generators. Each sphere was given a different material property with different colors of glass, metal, and plastic. The pedestal itself was texture mapped with a polished marble texture. The scene specification files are included on the companion DVD-ROM as `rnl_scene.rad` and `gensup.sh`.

9.2.3 MAP THE LIGHT PROBE TO AN EMISSIVE SURFACE SURROUNDING THE SCENE

In IBL, the scene is surrounded (either conceptually or literally) by a surface that the light probe image is mapped onto. In the simplest case, this surface is an infinite sphere. The RNL animation used a large but finite inward-pointing cube, positioned so that the bottom of the pedestal sat centered on the bottom of the cube (Figure 9.3). The light probe image was mapped onto the inner surfaces of

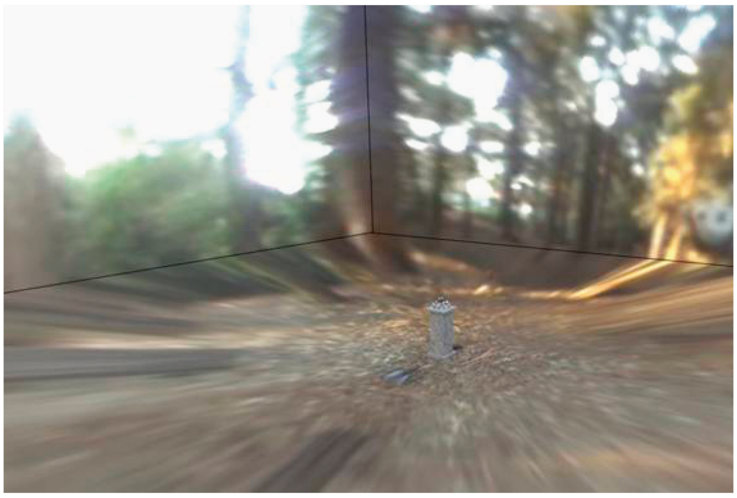


FIGURE 9.3 *The RNL pedestal is seen within the large surrounding box, texture mapped with the Eucalyptus Grove light probe image.*

the cube so that from the perspective of the top of the pedestal the light from the environment would come from substantially the same directions as it would have in the forest. The RADIANCE shading language was sufficiently general to allow this mapping to be specified in a straightforward manner. When a ray hits a surface, it reports the 3D point of intersection $P = (P_x, P_y, P_z)$ to user-supplied equations that compute the texture map coordinate for that point of the surface. In the RNL scene, the top of the pedestal was at the origin, and thus the direction vector into the probe was simply the vector pointing toward P. From this direction, the (u, v) coordinates for the corresponding pixel in the light probe image are computed using the angular map equations in Section 9.4.2. These calculations are specified in the file `angmap.cal` included on the companion DVD-ROM.

In IBL, the surface surrounding the scene is specified to be *emissive*, so that its texture is treated as a map of light emanating from the surface rather than a map of surface reflectance. In RADIANCE this is done by assigning this environment the *glow* material, which tells the renderer that once a ray hits this surface the radiance along the ray should be taken directly as the HDR color in the image, rather than the product of the texture color and the illumination incident upon it. When the environment surface is viewed directly (as in most of Figure 9.3), it appears as an image-based rendering with the same pixel colors as in the original light probe image.

9.2.4 RENDER THE SCENE AS ILLUMINATED BY THE IBL ENVIRONMENT

With the scene modeled and the light probe image mapped onto the surrounding surface, RADIANCE was ready to create the renderings using IBL. Appropriate rendering parameters were chosen for the number of rays to be used per pixel, and a camera path was animated to move around and within the scene. RADIANCE simulated how the objects would look as if illuminated by the light from the environment surrounding them. Some renderings from the resulting image sequence are shown in Figure 9.4. This lighting process is explained in further detail in Section 9.5.

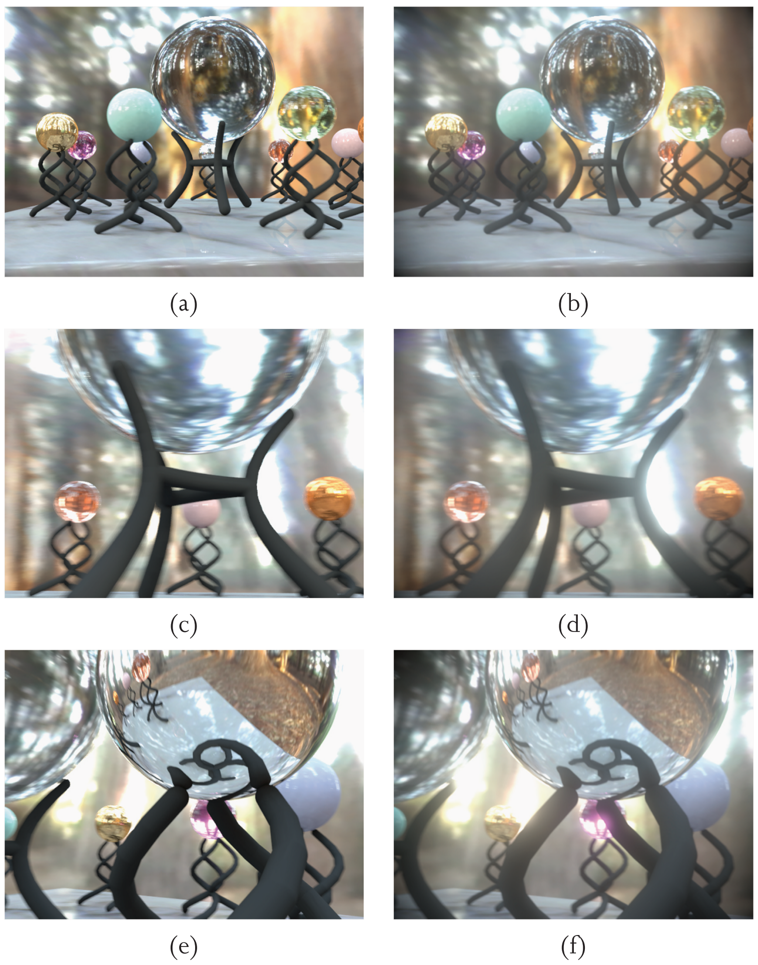


FIGURE 9.4 *Three frames from Rendering with Natural Light. Rendered frames (a, c, and e) before postprocessing. After postprocessing (b, d, and f) as described in Section 9.2.5.*

9.2.5 POSTPROCESS THE RENDERINGS

A final step in creating an IBL rendering or animation is to choose how to tone map the images for display. RADIANCE and many recent rendering systems can output their renderings as HDR image files. Because of this, the RNL renderings exhibited the full dynamic range of the original lighting environment, including the very bright areas seen in the sky and in the specular reflections of the glossy spheres. Because the renderings exhibit a greater dynamic range than can be shown on typical displays, some form of tone mapping is needed to produce the final displayable images. The most straightforward method of tone mapping is to pick a visually pleasing exposure factor for the image, truncate the bright regions to the maximum "white" value of the display, and apply any needed compensation for the response curve of the display (most commonly, applying a gamma-correction function). With this technique, values below the white point are reproduced accurately, but everything above the white point is clipped.

Chapters 6, 7, and 8 discussed several tone-reproduction operators that reduce the dynamic range of an image in a natural way to fall within the range of the display, any of which could be used for postprocessing IBL images. Another approach to postprocessing HDR images is to simulate some of the optical imperfections of a real camera system that communicate the full dynamic range of bright regions through blooming and vignetting effects. Such operators often work well in conjunction with IBL rendering because, like IBL, they are designed to simulate the appearance of photographic imagery.

Let's first examine how a blurring operator can communicate the full dynamic range of a scene even on an LDR display. The top of Figure 9.5 shows two bright square regions in an image. In the HDR image file, the right-hand square is six times brighter than the left (as seen in the graph below the squares). However, because the maximum "white" point of the display is below the brightness of the dimmer square the displayed squares appear to be the same intensity. If we apply a Gaussian blur convolution to the HDR pixel values, the blurred squares appear very different, even when clipped to the display's white point. The dim blurred square now falls considerably below the white point, whereas the middle of the bright blurred square still exceeds the range of the display. The brighter region also appears larger, even though the regions were originally the same size and are filtered with the same amount of blur. This effect is called *blooming*.

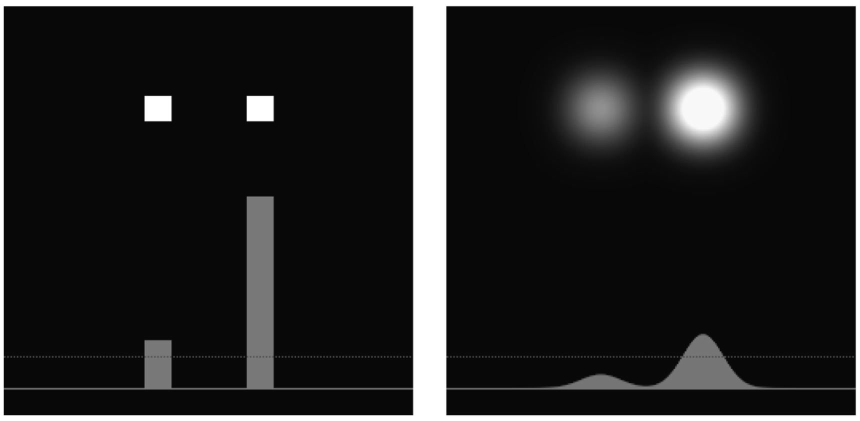

(a) Before blur (b) After blur

FIGURE 9.5 *Two bright image squares (a) with pixel values 1.5 and 6 are shown on a display with a white point of 1. Because of clipping, the squares appear the same. A graph of pixel intensity for a scan line passing through the centers of the squares is shown below, with "white" indicated by the dotted line. After applying a Gaussian blur (b), the blurred squares are very different in appearance, even though they are still clipped to the display white point.*

Similar blooming effects are seen frequently in real photographs, in which blur can be caused by any number of factors, including camera motion, subject motion, image defocus, "soft focus" filters placed in front of the lens, dust and coatings on lens surfaces, and scattering of light within the air and image sensor. Figure 9.6(a) shows an image acquired in HDR taken inside Stanford's Memorial church. When a clipped LDR version of the image is blurred horizontally (Figure 9.6(b)), the bright stained-glass windows become noticeably darker. When the HDR version of the image is blurred with the same filter (Figure 9.6(c)), the windows appear as vibrant bright streaks, even when clipped to the white point of the display. In addition to

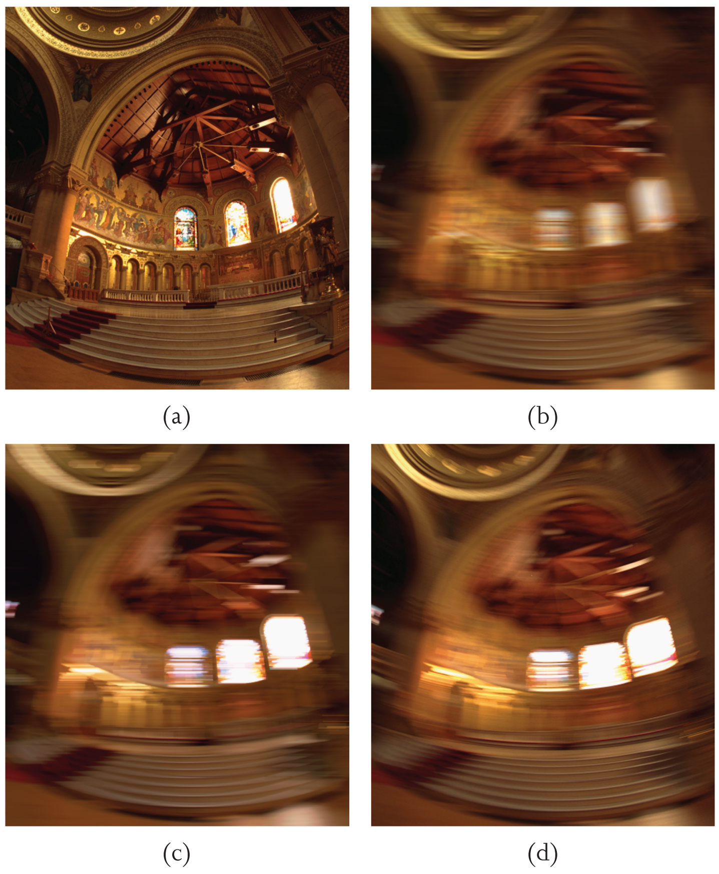
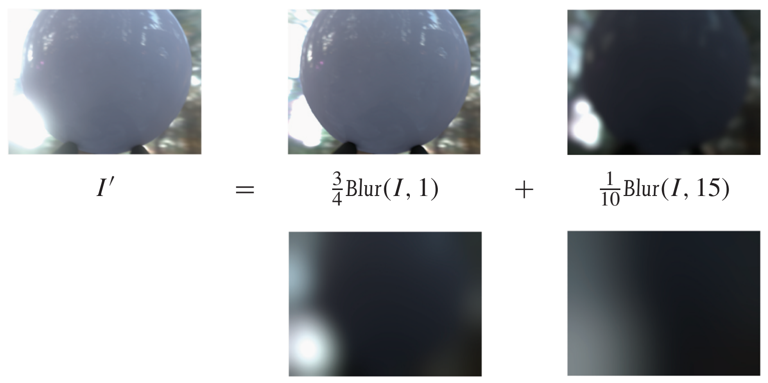


FIGURE 9.6 *An HDR scene inside a church with bright stained-glass windows. (b) Horizontally blurring the clipped LDR image gives the effect of image motion, but it noticeably darkens the appearance of the stained-glass windows. (c) Blurring an HDR image of the scene produces bright and well-defined streaks from the windows. (d) Real motion blur obtained by rotating camera during the exposure validates the HDR blurring simulated in image* **c**.

the HDR image series, the photographer also acquired a motion-blurred version of the church interior by rotating the camera on the tripod during a half-second exposure (Figure 9.6(d)). The bright streaks in this real blurred image (though not perfectly horizontal) are very similar to the streaks computed by the HDR blurring process seen in Figure 9.6(c), and dissimilar to the LDR blur seen in Figure 9.6(d).

The renderings for *Rendering with Natural Light* were postprocessed using a summation of differently blurred versions of the renderings produced by RADIANCE. Each final image used in the film was a weighted average of several differently blurred versions of the image. All of the blur functions used in RNL were Gaussian filters, and their particular mixture is illustrated in Figure 9.7.

With the right techniques, such blurring processes can be performed very efficiently, even though convolving images with wide filters is normally computationally expensive. First, Gaussian filters are *separable*, so that the filter can be performed



$$I' = \tfrac{3}{4}\mathrm{Blur}(I, 1) + \tfrac{1}{10}\mathrm{Blur}(I, 15) + \tfrac{1}{10}\mathrm{Blur}(I, 50) + \tfrac{1}{20}\mathrm{Blur}(I, 150)$$

FIGURE 9.7 *The mix of blur filters used to postprocess the RNL frames.* $\mathrm{Blur}(I, \sigma)$ *indicates a Gaussian blur with a standard deviation of* σ *pixels.*

as a 1D Gaussian blur in x followed by a 1D Gaussian blur in y. Second, a Gaussian blur can be closely approximated as several successive box blurs. Finally, the wider blurs can be closely approximated by applying correspondingly narrower blurring filters to a low-resolution version of the image, and then up-sampling the small blurred version. With these enhancements, efficient implementations of such techniques can be achieved on modern GPUs [187], and considerably more elaborate lens flare effects can be performed in real time [176].

Postprocessed frames from RNL can be seen in the right-hand column of Figure 9.4. Because the final postprocessed images are 75% composed of the original renderings with just a slight blur applied, the original image detail is still evident. However, because of the other blurred versions added, the bright parts of the environment and their specular reflections tend to bloom in the final renderings. Often, as in Figure 9.7, the bloom from bright spots in the environment appears to "wrap" around objects in the foreground. This is a surprisingly natural effect that helps the rendered objects appear to belong within the rest of the scene.

As mentioned previously, this postprocessing of HDR imagery is a form of tone mapping, the effects of which are similar to the results produced by using "soft focus," "mist," and "fog" filters on real camera lenses. The effects are also similar to the effects of the optical imperfections of the human eye. A detailed model of the particular glare and bloom effects produced in the human eye is constructed and simulated by Spencer et al. [198]. In addition, a basic model of human eye glare was used in conjunction with a tone-reproduction operator by Larson et al. [180].

A final subtle effect applied to the renderings in RNL is *vignetting*. Vignetting is the process of gently darkening the pixel values of an image toward its corners, which occurs naturally in many camera lenses (particularly at wide apertures) and is sometimes intentionally exaggerated with an additional mask or iris for photographic effect. Applying this effect to an HDR image before tone mapping the pixel values can also help communicate a greater sense of the range of light in a scene, particularly in animations. With this effect, as a bright region moves from the center of the field of view to the edge, the pixels around it dim. However, its particularly bright pixels will still reach the white point of the display. Thus, different exposures of the scene are revealed in a natural manner simply through camera motion. The effect is easily achieved by multiplying an image by a brightness falloff image such as in Figure 9.8(a). Figures 9.8(b) and 9.8(c) show a rendering from RNL before and after vignetting.

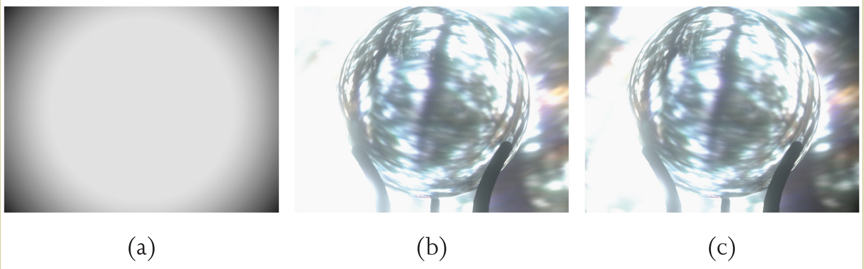


FIGURE 9.8 *Via the brightness fall-off function shown in image (a), a frame from the RNL animation is seen (b) before and (c) after vignetting. In this image, this operation darkens the right-side image corners, reveals detail in the slightly overexposed upper left corner, and has little effect on the extremely bright lower left corner. In motion, this vignetting effect helps communicate the HDR environment on an LDR display.*

As an early IBL example, RNL differed from most CG animations in that designing the lighting in the scene was a matter of choosing real light from a real location rather than constructing the light as an arrangement of computer-generated light sources. Using global illumination to simulate the image-based lighting naturally produced shading, highlights, refractions, and shadows that were consistent with one another and with the environment surrounding the scene. With traditional CG lighting, such an appearance would have been difficult to achieve.

9.3 CAPTURING LIGHT PROBE IMAGES

Moving on from our basic example, the next few sections present the key stages of IBL with greater generality and detail, beginning with the process of lighting capture. Capturing the incident illumination at a point in space requires taking an image with two properties. First, it must see in all directions, in that light coming from anywhere can affect the appearance of an object. Second, it must capture the full dynamic range of the light within the scene, from the brightest concentrated light sources to the dimmer but larger areas of indirect illumination from other surfaces in the scene. In many cases, the standard HDR photography techniques presented in Chapter 4 satisfy this second requirement. Thus, the remaining challenge is to acquire images that see in all directions, a process known as panoramic (or omnidirectional) photography. There are several methods of recording images that see in all directions, each with advantages and disadvantages. In this section, we describe some of the most common techniques, which include using mirrored spheres, tiled photographs, fish-eye lenses, and scanning panoramic cameras. This section concludes with a discussion of how to capture light probe images that include a direct view of the sun, which is usually too bright to record with standard HDR image capture techniques.

9.3.1 PHOTOGRAPHING A MIRRORED SPHERE

The technique used to capture the *Rendering with Natural Light* light probe was to photograph a mirrored sphere placed in the scene where it is desired to capture the illumination. Using mirrored spheres to obtain omnidirectional reflections of an environment was first used for *environment mapping* [186,207,171] (described in Section 9.8.1), where such images are directly texture mapped onto surfaces of objects. The main benefit of photographing a mirrored sphere is that it reflects very nearly the entire environment in a single view. Aside from needing two tripods (one for the sphere and one for the camera), capturing a light probe image with this technique can be fast and convenient. Mirrored spheres are inexpensively available as 2- to 4-inch diameter chrome ball bearings (available from the McMaster–Carr catalog, *www.mcmaster.com*), 6- to 12-inch mirrored glass lawn ornaments (available from Baker's Lawn Ornaments, *www.bakerslawnorn.com*), and Chinese meditation balls (1.5 to 3 inches). Dubé juggling equipment (*www.dube.com*) sells polished hollow chrome spheres from 2-1/6 to 2-7/8 inches in diameter. Professionally manufactured mirrored surfaces with better optical properties are discussed at the end of this section.

There are several factors that should be considered when acquiring light probe images with a mirrored sphere. These are discussed in the sections that follow.

Framing and Focus First, it is desirable to have the sphere be relatively far from the camera to minimize the size of the camera's reflection and to keep the view nearly orthographic. To have the sphere be relatively large in the frame, it is necessary to use a long-focal-length lens. Many long lenses have difficulty focusing closely on small objects, and thus it may be necessary to use a +1 diopter close-up filter (available from a professional photography store) on the lens to bring the sphere into focus. The image of the sphere usually has a shallow depth of field, especially when a close-up filter is used, and thus it is often necessary to use an aperture of f/8 or smaller to bring the full image into focus.

Blind Spots There are several regions in a scene that are usually not captured well by a mirrored sphere. One is the region in front of the sphere, which reflects the camera and often the photographer. Another is the region directly behind the sphere, which is reflected by a thin area around the sphere's edge. The last is a strip of area from straight down and connecting to the area straight back, which usually reflects whatever supports the sphere. For lighting capture, these effects are easily minimized by orienting the camera so that no photometrically interesting areas of the scene (e.g., bright light sources) fall within these regions. However, it is sometimes desirable to obtain clear images of all directions in the environment, for example when the light probe image itself will be seen in the background of the scene. To do this, one can take two HDR images of the mirrored sphere, with the second rotated 90 degrees around from the first. In this way, the poorly represented forward and backward directions of one sphere correspond to the well-imaged left and right directions of the other, and vice versa. The two images taken for the RNL light probe are shown in Figure 9.9. Each image slightly crops the top and bottom of the sphere, which was done intentionally to leverage the fact that these areas belong to the rear half of the environment that appears in the other sphere image. Using an HDR image-editing program such as HDR Shop [202], these two images can be combined into a single view of the entire environment that represents all directions well except straight down. If needed, this final area could be filled in from a photograph of the ground or through manual image editing.

Calibrating Sphere Reflectivity It is important to account for the fact that mirrored spheres are generally not optically perfect reflectors. Though the effect is often unnoticed, ball bearings typically reflect only a bit more than half of the light hitting

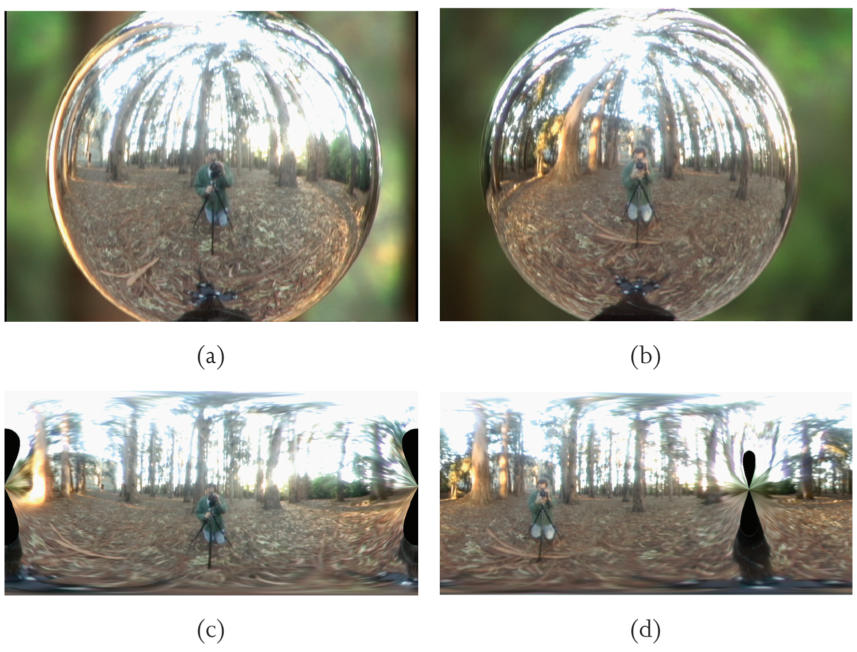


FIGURE 9.9 *Acquiring a light probe with a mirrored sphere. (a) and (b) show two images of the sphere, taken 90 degrees apart. (c) and (d) show the mirrored sphere images transformed into latitude-longitude mappings. In (d), the mapping has been rotated 90 degrees to the left to line up with the mapping of (c). The black teardrop shapes correspond to the cropped regions at the top and bottom of each sphere, and the pinched area between each pair of drops corresponds to the poorly sampled region near the outer edge of the sphere image. Each sphere yields good image data where the other one has artifacts, and combining the best regions of each can produce a relatively seamless light probe image, as in Figure 9.2.*

them. In some IBL applications, the lighting is captured using a mirrored sphere, and the background image of the scene is photographed directly. To correct the sphere image so that it is photometrically consistent with the background, we need to measure the reflectivity of the sphere. This can be done using a setup such as that shown in Figure 9.10. In this single photograph, taken with a radiometrically calibrated camera, the indicated patch of diffuse paper is reflected in the ball. We can thus divide the average pixel value of the patch in the reflection by the average pixel value in the direct view to obtain the sphere's percent reflectivity in each of the red, green, and blue channels. A typical result would be $(0.632, 0.647, 0.653)$. Often, these three numbers will be slightly different, indicating that the sphere tints the incident light. The light probe image can be corrected to match the background by dividing its channels by each of these numbers.

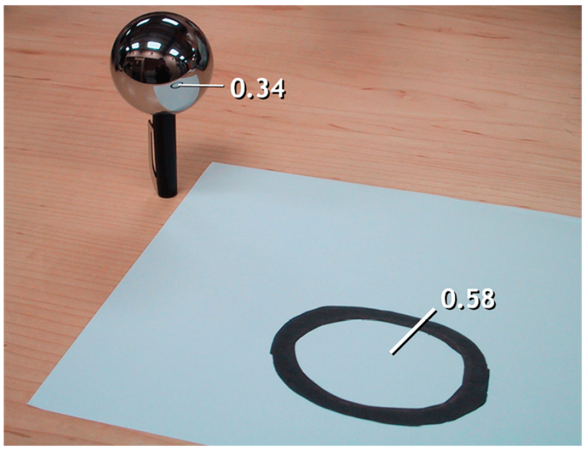


FIGURE 9.10 *The reflectivity of a mirrored sphere can be determined by placing the sphere near an easy-to-identify part of a diffuse surface, and comparing its brightness seen directly to its appearance as reflected in the sphere. In this case, the sphere is 59% reflective.*

Nonspecular Reflectance Mirrored spheres usually exhibit a faint diffuse or rough specular component due to microscopic scratches and deposits on their surface. It is best to keep the spheres in cloth bags to minimize such scratching, as well as to keep them dry so that the surface does not oxidize. A slight rough specular component usually makes little difference in how light probe images illuminate CG objects, but when viewed directly the reflected image may lack contrast in dark regions and exhibit bright flares around the light sources. If an application requires a near-perfectly shiny mirrored surface, one can have a glass or metal sphere or hemisphere specially coated with a thin layer of aluminum by an optical coating company (only half a sphere can be photographed at once, and thus in practice a hemisphere can also be used as a light probe). Such coated optics yield extremely clear specular reflections and can be up to 91% reflective. Some experiments in capturing the sky using such optics can be found in Stumpfel [199].

Polarized Reflectance Mirrored spheres behave somewhat unexpectedly with respect to polarization. Light that reflects from a sphere at angles next to the outer rim becomes polarized, an effect characterized by Fresnel's equations [168]. Camera sensors generally record light irrespective of polarization, so this itself is not a problem.[2] However, for the same reasons, polarized light reflecting from a mirrored sphere can appear either too bright or too dim compared to being viewed directly. This is a significant effect in outdoor environments, where the scattered blue light of the sky is significantly polarized. This problem can be substantially avoided by using highly reflective mirror coatings (as discussed previously).

Image Resolution It can be difficult to obtain a particularly high-resolution image of an environment as reflected in a mirrored sphere, because only one image is used to cover a fully spherical field of view. For lighting CG objects, the need for highly detailed light probe images is not great: only large and very shiny objects reflect light in a way that fine detail in an environment can be noticed. However, for forming virtual backgrounds behind CG objects it is often desirable to have higher-resolution imagery. In the RNL animation, the low resolution of the light probe image used

2 Whereas sensors tend to detect different polarization directions equally, wide-angle lenses can respond differently according to polarization for regions away from the center of the image.

for the background produced the reasonably natural appearance of a shallow depth of field, even though no depth of field effects were simulated.

Photographing a mirrored sphere is an example of a *catadioptric* imaging system in that it involves both a lens and a reflective surface. In addition to mirrored spheres, other shapes can be used that yield different characteristics of resolution, depth of field, and field of view. One example of a well-engineered omnidirectional video camera covering slightly over a hemispherical field of view is presented in Nayer [188]. Nonetheless, for capturing illumination wherein capturing the full sphere is more important than high image resolution, mirrored spheres are often the most easily available and convenient method.

9.3.2 TILED PHOTOGRAPHS

Omnidirectional images can also be captured by taking a multitude of photographs looking in different directions and "stitching" them together — a process made familiar by QuickTime VR panoramas [156]. This technique can be used to assemble remarkably high-resolution omnidirectional images using a standard camera and lens. Unfortunately, the most commonly acquired panoramas see all the way around the horizon but only with a limited vertical field of view. For capturing lighting, it is important to capture imagery looking in *all* directions, particularly upward, because this is often where much of the light comes from. Images taken to form an omnidirectional image will align much better if the camera is mounted on a nodal rotation bracket, which can eliminate viewpoint parallax between the various views of the scene. Such brackets are available commercially from companies such as Kaidan (*www.kaidan.com*). Some models allow the camera to rotate around its nodal center for both the horizontal and vertical axes.

Figure 9.11 shows tone-mapped versions of the source HDR images for the "Uffizi Gallery" light probe image [160], which was acquired as an HDR tiled panorama. These images were aligned by marking pairs of corresponding points between the original images and then solving for the best 3D rotation of each image to minimize the distance between the marked points. The images were then blended across their edges to produce the final full-view latitude-longitude mapping. This image was used as the virtual set and lighting environment for the middle sequence of the animation *Fiat Lux*.

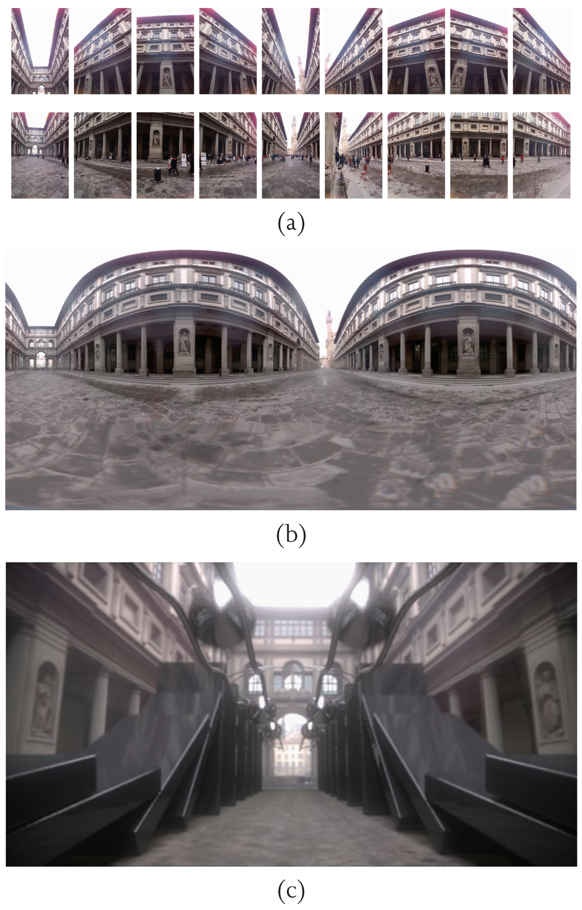


FIGURE 9.11 (a) The Uffizi light probe was created from an HDR panorama with two rows of nine HDR images. (b) The assembled light probe in latitude-longitude format. (c) Synthetic objects added to the scene, using the light probe as both the virtual background and the lighting environment.

A more automatic algorithm for assembling such full-view panoramas is described in [201], and commercial image-stitching products such as QuickTime VR (*www.apple.com/quicktime/qtvr/*) and Realviz Stitcher (*www.realviz.com*) allow one to interactively align images taken in different directions to produce full-view panoramas in various image formats. Unfortunately, at the time of writing no commercial products natively support stitching HDR images. Digital photographer Greg Downing (*www.gregdowning.com*) has described a process [166] for stitching each set of equivalent exposures across the set of HDR images into its own panorama, and then assembling this series of LDR panoramas into a complete light probe image. The key is to apply the same alignment to every one of the exposure sets: if each set were aligned separately, there would be little chance of the final stitched panoramas aligning well enough to be assembled into an HDR image. To solve this problem, Downing uses Realviz Stitcher to align the exposure level set with the most image detail and saves the alignment parameters in a way that they can be applied identically to each exposure level across the set of views. These differently exposed LDR panoramas can then be properly assembled into an HDR panorama.

9.3.3 FISH-EYE LENSES

Fish-eye lenses are available for most single-lens reflex cameras and are capable of capturing 180 degrees or more of an environment in a single view. As a result, they can cover the full view of an environment in as few as two images. In Greene [171], a fish-eye photograph of the sky was used to create the upper half of a cube map used as an environment map. Although fish-eye lens images are typically not as sharp as regular photographs, light probe images obtained using fish-eye lenses are usually of higher resolution than those obtained by photographing mirrored spheres. A challenge in using fish-eye lenses is that not all 35-mm digital cameras capture the full field of view of a 35-mm film camera due to having a smaller image sensor. In this case, the top and bottom of the circular fish-eye image are usually cropped off. This can require taking additional views of the scene to cover the full environment. Fortunately, recently available digital cameras, such as the Canon EOS 1Ds and the Kodak DCS 14n, have image sensor chips that are the same size as 35-mm film (and no such cropping occurs).

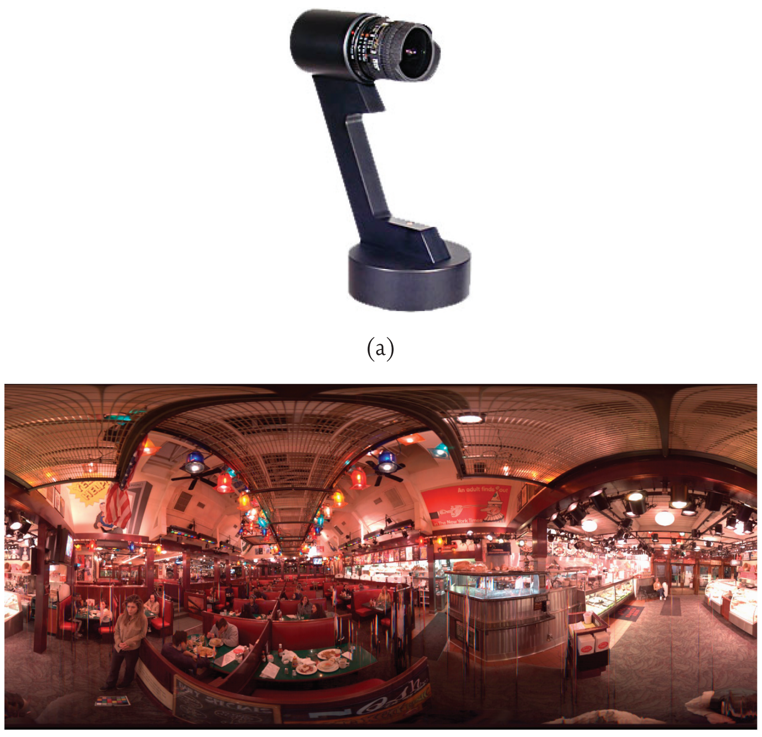
Fish-eye lenses can exhibit particularly significant radial intensity fall-off, also known as *vignetting*. As with other lenses, the amount of falloff tends to increase with the size of the aperture being used. For the Sigma 8-mm fish-eye lens, the amount of falloff from the center to the corners of the image is more than 50% at its widest aperture of f/4. The falloff curves can be calibrated by taking a series of photographs of a constant intensity source at different camera rotations and fitting a function to the observed image brightness data. This surface can be used to render an image of the *flat-field response* of the camera. With this image, any HDR image obtained with the camera can be made radiometrically consistent across its pixels by dividing by the image of the flat-field response. An example of this process is described in more detail in [200].

9.3.4 SCANNING PANORAMIC CAMERAS

Scanning panoramic cameras are capable of capturing particularly high-resolution omnidirectional HDR images. These cameras use narrow image sensors that are typically 3 pixels wide and several thousand pixels tall. The three columns of pixels are filtered by red, green, and blue filters, allowing the camera to sense color. A precise motor rotates the camera by 360 degrees over the course of a few seconds to a few minutes, capturing a vertical column of the panoramic image many times per second. When a fish-eye lens is used, the full 180-degree vertical field of view can be captured from straight up to straight down. Two cameras based on this process are made by Panoscan and Spheron VR (Figure 9.12).

Trilinear image sensors, having far fewer pixels than area sensors, can be designed with more attention given to capturing a wide dynamic range in each exposure. Nonetheless, for IBL applications in which it is important to capture the full range of light, including direct views of concentrated light sources, taking multiple exposures is usually still necessary. The Panoscan camera's motor is able to precisely rewind and repeat its rotation, allowing multiple exposures to be taken and assembled into an HDR image without difficulty. The Spheron VR camera (see also Section 4.9.4) can be ordered with a special HDR feature in which the image sensor rapidly captures an HDR series of exposures for each column of pixels as the camera head rotates. These differently exposed readings of each pixel column can be assembled into HDR images from just one rotation of the camera.

For these cameras, the speed of scanning is limited by the amount of light in the scene. Suppose that at the chosen f-stop and ISO setting it takes $\frac{1}{125}$ of a second to obtain a proper exposure of the shadows and midtones of a scene. If the


(a)

(b)

FIGURE 9.12 *(a) The Spheron scanning panoramic camera. (b) A tone-mapped version of a high-resolution omnidirectional HDR image taken with the camera. Panorama courtesy of Ted Chavalas of Panoscan, Inc.*

image being acquired is 12,000 pixels wide, the camera must take at least 96 seconds to scan the full panorama. Fortunately, shooting the rest of the HDR image series to capture highlights and light sources takes considerably less time because each shutter speed is shorter, typically at most one-fourth the length for each additional exposure. Although capturing lighting with scanning panoramic cameras is not instantaneous, the resulting lighting environments can have extremely detailed resolution. The high resolution also enables using these images as background plates or to create image-based models for 3D virtual sets.

9.3.5 CAPTURING ENVIRONMENTS WITH VERY BRIGHT SOURCES

For image-based lighting, it is important that the acquired light probe images cover the full dynamic range of light in the scene up to and including light sources. If 99.99% of a light probe image is recorded properly but 0.01% of the pixel values are saturated, the light captured could still be very inaccurate depending on how bright the saturated pixels really should have been. Concentrated light sources are often significantly brighter than the average colors within a scene. In a room lit by a bare light bulb, the light seen reflecting from tens of square meters of ceiling, floor, and walls originates from just a few square millimeters of light bulb filament. Because of such ratios, light sources are often thousands, and occasionally millions, of times brighter than the rest of the scene.

In many cases, the full dynamic range of scenes with directly visible light sources can still be recovered using standard HDR photography techniques, in that camera shutter speeds can usually be varied down to $\frac{1}{4000}$ of a second or shorter, and small apertures can be used as well. Furthermore, modern lighting design usually avoids having extremely concentrated lights (such as bare filaments) in a scene, preferring to use globes and diffusers to more comfortably spread the illumination over a wider area. However, for outdoor scenes the sun is a light source that is both very bright and very concentrated. When the sun is out, its brightness can rarely be recorded using a typical camera even using the shortest shutter speed, the smallest aperture, and the lowest sensor gain settings. The sun's brightness often exceeds that of the sky and clouds by a factor of fifty thousand, which is difficult to cover using varying shutter speeds alone.

Stumpfel et al. [200] presented a technique for capturing light from the sky up to and including the sun. To image the sky, the authors used a Canon EOS 1Ds digital camera with a Sigma 8-mm fish-eye lens facing upward on the roof of an office building (Figure 9.13(a)). The lens glare caused by the sun was minor, which was verified by photographing a clear sky twice and blocking the sun in one of the images. For nearly the entire field of view, the pixel values in the sky were within a few percent points of each other in both images.

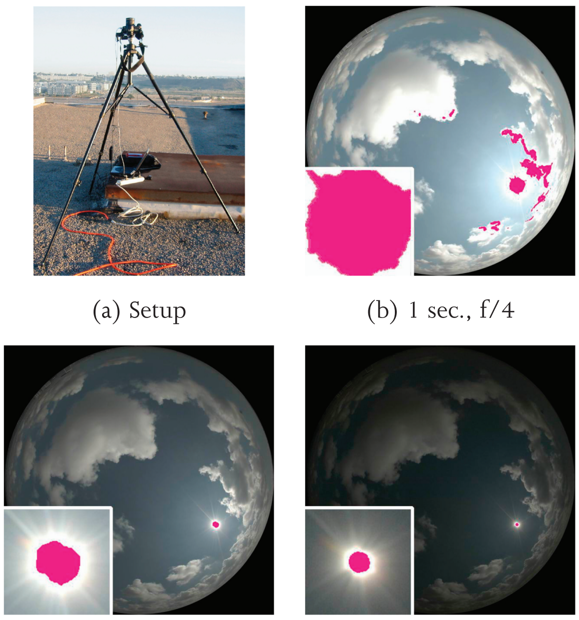
As expected, the sun was far too bright to record even using the camera's shortest shutter speed of $\frac{1}{8000}$ of a second at f/16, a relatively small aperture.[3] The authors thus placed a Kodak Wratten 3.0 neutral density (ND) filter on the back of the lens to uniformly reduce the light incident on the sensor by a factor of one thousand.[4] ND filters are often not perfectly neutral, giving images taken though them a significant color cast. The authors calibrated the transmission of the filter by taking HDR images of a scene with and without the filter, and divided the two images to determine the filter's transmission in the red, green, and blue color channels. All images subsequently taken through the filter were scaled by the inverse of these transmission ratios.

Having the 3.0 ND filter on the lens made it possible to image the sun at $\frac{1}{8000}$ of a second at f/16 without saturating the sensor (see Figure 9.13(a)), but it made the sky and clouds require an undesirably long exposure time of 15 seconds. To solve this problem, the authors used a laptop computer to control the camera so that both the shutter speed and the aperture could be varied during each HDR image sequence. Thus, the series began at f/4 with exposures of 1, $\frac{1}{4}$, and $\frac{1}{32}$ second and then switched to f/16 with exposures of $\frac{1}{16}$, $\frac{1}{125}$, $\frac{1}{1000}$, and $\frac{1}{8000}$ of a second. Images from such a sequence are seen in Figures 9.13(b) through 9.13(h). For pre-sunrise and post-sunset images, the f/16 images were omitted and an additional exposure of 4 seconds at f/4 was added to capture the dimmer sky of dawn and dusk.

Creating HDR images using images taken with different apertures is slightly more complicated than usual. Because different apertures yield different amounts of lens vignetting, each image needs to be corrected for its aperture's flat-field response (see Section 9.3.3) before the HDR assembly takes place. In addition, whereas the actual exposure ratios of different camera shutter speeds typically follow the expected geometric progression ($\frac{1}{30}$ second is usually precisely half the exposure of

3 The authors observed unacceptably pronounced star patterns around the sun at the smaller apertures of f/22 and f/32 due to diffraction of light from the blades of the iris.

4 Because the fish-eye lens has such a wide-angle view, filters are placed on the back of the lens using a small mounting bracket rather than on the front.

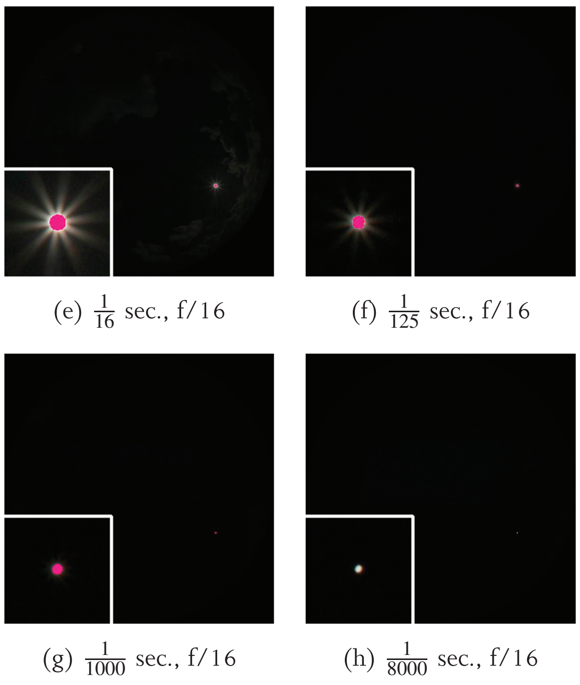


(a) Setup (b) 1 sec., f/4

(c) $\frac{1}{4}$ sec., f/4 (d) $\frac{1}{32}$ sec., f/4

FIGURE 9.13 *(a) A computer-controlled camera with a fish-eye lens placed on a roof to capture HDR images of the sky. A color-corrected HDR image series (b through h) of a sunny sky taken using varying shutter speed, aperture, and a 3.0 neutral density filter. The inset of each frame shows a small area around the sun, with saturated pixels shown in pink. Only the least exposed image (h) accurately records the sun's intensity without sensor saturation.*

$\frac{1}{15}$ second; $\frac{1}{15}$ second is usually precisely half the exposure of $\frac{1}{8}$ second), aperture transmission ratios are less exact. In theory, images taken at f/4 should receive sixteen times the exposure of images taken at f/16, but generally do not.



(e) $\frac{1}{16}$ sec., f/16 (f) $\frac{1}{125}$ sec., f/16

(g) $\frac{1}{1000}$ sec., f/16 (h) $\frac{1}{8000}$ sec., f/16

FIGURE 9.13 *(continued)*

To test this, the authors took images of a constant intensity light source at both f/4 and f/16 and compared the pixel value ratios at the center of the image, measuring a factor of 16.63 rather than 16, and compensated for this ratio accordingly.

When the sun was obscured by clouds or was low in the sky, the images with the shortest exposure times were not required to cover the full dynamic range.

The authors discovered they could adaptively shoot the HDR image sequence by programming the laptop computer to analyze each image in the series after it was taken. If one of the images in the series was found to have no saturated pixels, no additional photographs were taken.

Indirectly Deriving Sun Intensity from a Diffuse Sphere Rather than using specialized HDR photography to directly capture the sun's intensity, we can estimate the sun's intensity based on the appearance of a diffuse object within the scene. In particular, we can use a mirrored sphere to image the ground, sky, and clouds and a diffuse sphere to indirectly measure the intensity of the sun. Such an image pair is shown in Figure 9.14.

The mirrored sphere contains accurate pixel values for the entire sky and surrounding environment, except for the region of sensor saturation near the sun. Because of this, the clipped light probe will not accurately illuminate a synthetic object as it would really appear in the real environment; it would be missing the light from the sun. We can quantify the missing light from the sun region by comparing the real and synthetic diffuse sphere images.

To perform this comparison, we need to adjust the images to account for each sphere's reflectivity. For the diffuse sphere, a gray color can be preferable to white because it is less likely to saturate the image sensor when exposed according to the average light in the environment. The paint's reflectivity can be measured by painting a flat surface with the same paint and photographing this sample (with a radiometrically calibrated camera) in the same lighting and orientation as a flat surface of known reflectance. Specialized *reflectance standards* satisfying this requirement are available from optical supply companies. These reflect nearly 99% of the incident light — almost perfectly white. More economical reflectance standards are the neutral-toned squares of a Gretag–MacBeth ColorChecker chart (*www.gretagmacbeth.com*), whose reflectivities are indicated on the back of the chart. Let us call the reflectivity of our standard ρ_{standard}, the pixel color of the standard in our image L_{standard}, and the pixel color of the paint sample used for the diffuse sphere L_{paint}. Then the reflectivity ρ_{paint} of the paint is simply:

$$\rho_{\text{paint}} = \rho_{\text{standard}} \frac{L_{\text{paint}}}{L_{\text{standard}}}.$$

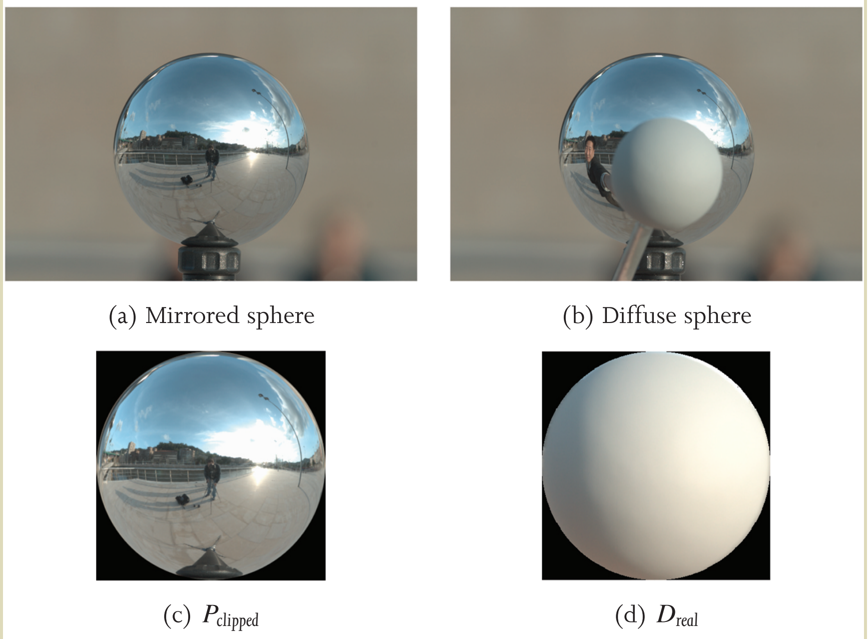


FIGURE 9.14 *(a) A mirrored sphere photographed as a single LDR exposure with direct sunlight. The bright sun becomes a saturated region of pixels clipped to white. (b) A gray diffuse sphere photographed in the same illumination, held several inches in front of the mirrored sphere. (c) The cropped, reflectance-calibrated, and clipped mirrored sphere image* P_{clipped}*. (d) The cropped and reflectance-calibrated image of the diffuse sphere* D_{real}*.*

For the diffuse paint used to paint the sphere in Figure 9.14(b), the reflectivity was measured to be (0.320, 0.333, 0.346) in the red, green, and blue channels, indicating that the chosen paint is very slightly bluish. Dividing the pixel values of the diffuse sphere by its reflectivity yields the appearance of the diffuse sphere as if it were 100% reflective, or perfectly white; we call this image D_{real} (see Figure 9.14(d)).

Likewise, the image of the mirrored sphere should be divided by the mirrored sphere's reflectivity as described in Section 9.3.1 and Figure 9.10 to produce the image that would have been obtained from a perfectly reflective mirrored sphere.

With both sphere images calibrated, we can use IBL to light a synthetic white sphere with the clipped lighting environment image, which we call D_{clipped} in Figure 9.15. As expected, this synthetic sphere appears darker than the real sphere in the actual lighting environment D_{real}. If we subtract D_{clipped} from D_{real}, we obtain an image of the missing reflected light from the sun region (which we call D_{sun}). This operation leverages the additive property of light, described in detail in Section 9.9.

From the previous section we know that the sun is a 0.53-degree disk in the sky. To properly add such a source to P_{clipped}, it should be placed in the right direction and assigned the correct radiant intensity. The direction of the sun can usually be estimated from the clipped light probe image as the center of the saturated region in the image. The pixel coordinates (u, v) in this image can be converted to a direction vector (D_x, D_y, D_z) using the ideal sphere-mapping formula discussed in Section 9.4.1. If the saturated region is too large to locate the sun with sufficient precision, it is helpful to have at least one additional photograph of the mirrored

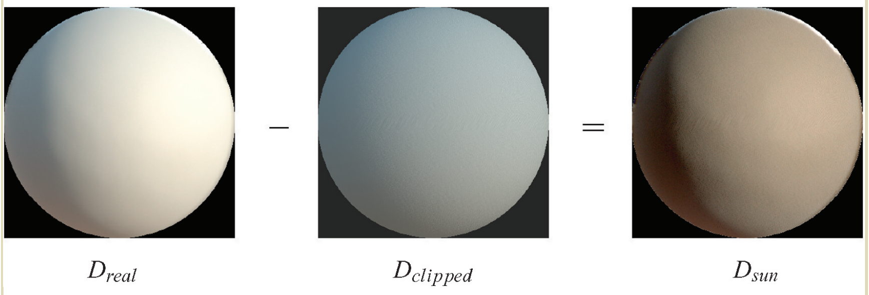


FIGURE 9.15 *The calibrated image of the real diffuse sphere D_{real}, minus the image of a synthetic diffuse sphere lit by the clipped IBL environment D_{clipped}, yields an image of the missing light from the sun D_{sun}.*

sphere taken with a shorter exposure time, or a photograph of a black plastic sphere in which the sun's position can be discerned more accurately. In this example, the sun's direction vector was measured as $(0.748, 0.199, 0.633)$ based on its position in the mirrored sphere image.

To determine the sun's radiance, we know that if the sun alone were to illuminate a white diffuse sphere the sphere should appear similar to D_{sun}. We begin by creating a sun disk with direction $(0.748, 0.199, 0.633)$, an angular extent of 0.53 degrees diameter, and an experimental radiance value of $L_{\mathrm{suntest}} = 46{,}700$. The specification for such a light source in a rendering file might look as follows.

```
light sun directional {
  direction 0.748  0.199  0.633
  angle 0.5323
  color 46700 46700 46700
}
```

The value 46,700 is chosen for convenience to be the radiant intensity of a 0.53 degrees diameter infinite light source that illuminates a diffuse white sphere such that its brightest spot (pointing toward the source) has a radiance of 1. Lighting a white sphere with this source produces the image D_{suntest} seen in Figure 9.16.

From the equation shown in Figure 9.16, we can solve for the unknown color α that best scales D_{suntest} to match D_{sun}. This is easily accomplished by dividing the average pixel values of the two images: $\alpha = \mathit{avg}(D_{\mathrm{sun}})/\mathit{avg}(D_{\mathrm{suntest}})$. Then, we can compute the correct radiant intensity of the sun as $L_{\mathrm{sun}} = \alpha L_{\mathrm{suntest}}$. For this example, applying this procedure to each color channel yields $\alpha = (1.166, 0.973, 0.701)$, which produces $L_{\mathrm{sun}} = (54500, 45400, 32700)$. Replacing the L_{suntest} value in the directional light specification file with this new L_{sun} value produces a directional light that models the missing sunlight in the clipped light probe image.

We can validate the accuracy of this procedure by lighting a diffuse sphere with the combined illumination from the clipped probe P_{clipped} and the reconstructed sun. Figures 9.17(a) and (b) show a comparison between the real diffuse sphere D_{real} and a synthetic diffuse sphere illuminated by the recovered environment. Subtracting (a) from (b) (shown in Figure 9.17(c)) allows us to visually and quantitatively verify the accuracy of the lighting reconstruction. The difference image is nearly black, which indicates a close match. In this case, the root mean squared intensity

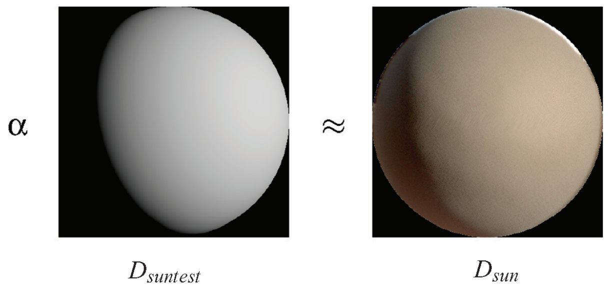


FIGURE 9.16 Lighting a white sphere (D_{suntest}) from the direction of the sun with an experimental radiant intensity L_{suntest} sets up an equation to determine the missing light from the sun.

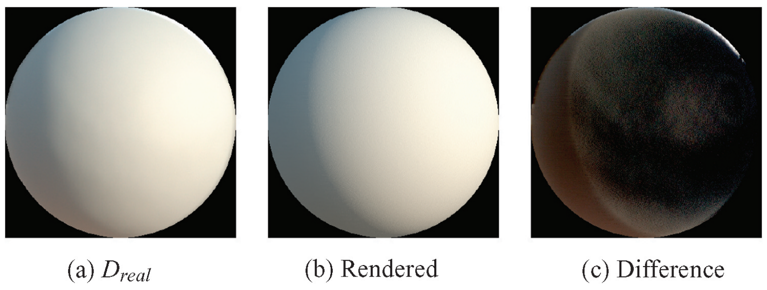


FIGURE 9.17 (a) The calibrated diffuse sphere D_{real} in the lighting environment. (b) A rendered diffuse sphere, illuminated by the incomplete probe P_{clipped} and the recovered sun. (c) The difference between the two. The nearly black image indicates that the lighting environment was recovered accurately.

difference between the real and simulated spheres is less than 2%, indicating a close numeric match as well. The area of greatest error is in the lower left of the difference image, which is a dim beige color instead of black due to unintended bounced light on the real sphere from the person's hand holding the sphere.

At this point, the complete lighting environment is still divided into two pieces: the clipped IBL environment and the directional sun source. To unify them, the sun disk could be rasterized into a mirror reflection image space and then added to the clipped IBL environment. However, as we will see in Section 9.6.1, it can be a great benefit to the rendering process if concentrated lights such as the sun are simulated as direct light sources rather than as part of the IBL environment (because of the sampling problem). Figure 9.38(e) later in this chapter shows a rendering of a collection of CG objects illuminated by the combination of the separate clipped probe and the direct sun source. The appearance of the objects is realistic and consistent with their environment. The rendering makes use of an additional technique (described in Section 9.7) to have the objects cast shadows onto the scene as well.

Images with concentrated light sources such as the sun not only require additional care to acquire but also pose computational challenges for image-based lighting algorithms. Section 9.6 describes why these challenges occur and how they can be solved through importance sampling techniques. Before beginning that topic, we will describe some of the omnidirectional image-mapping formats commonly used to store light probe images.

9.4 OMNIDIRECTIONAL IMAGE MAPPINGS

Once a light probe image is captured, it needs to be stored in an image file using an omnidirectional image mapping. This section describes four of the most commonly used image mappings and provides formulas to determine the appropriate (u, v) coordinates in the image corresponding to a unit direction in the world $D = (D_x, D_y, D_z)$, and vice versa. These formulas all assume a right-handed coordinate system in which $(0, 0, -1)$ is forward, $(1, 0, 0)$ is right, and $(0, 1, 0)$ is up.

This section also discusses some of the advantages and disadvantages of each format. These considerations include the complexity of the mapping equations, how much distortion the mapping introduces, and whether the mapping has special features that facilitate computing properties of the image or using it with specific rendering algorithms. The mappings this section presents are the *ideal mirrored sphere*, the *angular map*, *latitude-longitude*, and the *cube map*.

9.4.1 IDEAL MIRRORED SPHERE

For the ideal mirrored sphere we use a circle within the square image domain of $u \in [0, 1]$, $v \in [0, 1]$. The mapping equation for world to image is as follows.

$$r = \frac{\sin(\frac{1}{2}\arccos(-D_z))}{2\sqrt{{D_x}^2 + {D_y}^2}}$$

$$(u, v) = \left(\frac{1}{2} + rD_x\,,\ \frac{1}{2} - rD_y\right)$$

The mapping equation for image to world is as follows.

$$r = \sqrt{(2u-1)^2 + (2v-1)^2}$$

$$(\theta, \phi) = (\operatorname{atan2}(2u-1, -2v+1), 2\arcsin(r))$$

$$(D_x, D_y, D_z) = (\sin\phi\cos\theta, \sin\phi\sin\theta, -\cos\phi)$$

The ideal mirrored sphere mapping (Figure 9.18(a)) is how the world looks when reflected in a mirrored sphere, assuming an orthographic camera and a world that is distant relative to the sphere's diameter. In practice, real spheres exhibit a mapping similar to this ideal one as long as the sphere is small relative to its distance to the camera and to the objects in the environment. Of course, the reflection in a sphere is actually a mirror image of the environment, and thus the image should be flipped in order to be consistent with directions in the world.

Like all mappings in this section, the ideal mirrored sphere reflects all directions of the environment. Straight forward—that is, $D = (0, 0, -1)$—appears in the center of the image. Straight right, $D = (1, 0, 0)$, appears $\frac{\sqrt{2}}{2}$ of the way from the center to the right-hand edge of the image. Straight up appears $\frac{\sqrt{2}}{2}$ of the way from the center to the top of the image. Straight back is the one direction that

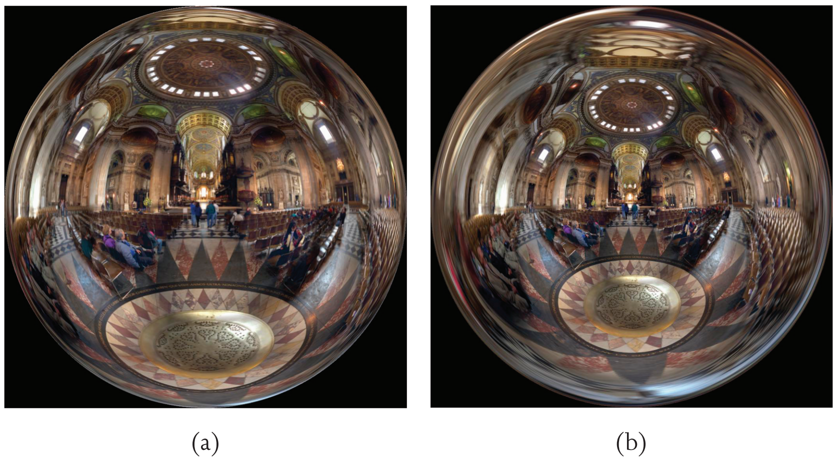


FIGURE 9.18 *(a) The ideal mirrored sphere mapping. (b) the angular map mapping. The mappings appear similar, but devote different amounts of image area to the front versus the back half of the environment.*

does not cleanly map to a particular image coordinate, and corresponds to the outer circumference of the circle's edge.

From these coordinates, we see that the front half of the environment is contained within a disk that is $\frac{\sqrt{2}}{2}$ of the diameter of the full image. This makes the area taken up by the front half of the environment precisely equal to the area taken up by the back half of the environment. This property generalizes in that any two regions of equal solid angle in the scene will map to the same amount of area in the image. One use of this equal-area property is to calculate the average illumination color in a light probe (the average pixel value within the image circle is the average value in the environment). If the image has a black background behind the circular image, the average value in the scene is $\frac{4}{\pi}$ of the average of the entire square.

A significant disadvantage with the mirrored sphere mapping is that the back half of the environment becomes significantly stretched in one direction and squeezed in the other. This problem increases in significance toward the outer edge of the circle, becoming extreme at the edge. This can lead to the same problem we saw with mirrored sphere light probe images, in that the regions around the edge are poorly sampled in the radial direction. Because of this, the mirrored sphere format is not a preferred format for storing omnidirectional images, and the angular map format is frequently used instead.

9.4.2 ANGULAR MAP

For the angular map we also use a circle within the square image domain of $u \in [0, 1]$, $v \in [0, 1]$. The mapping equation for world to image is as follows.

$$r = \frac{\arccos(-D_z)}{2\pi\sqrt{{D_x}^2 + {D_y}^2}}$$

$$(u, v) = \left(\frac{1}{2} - rD_y, \frac{1}{2} + rD_x\right)$$

The equation for image to world is as follows.

$$(\theta, \phi) = \left(\operatorname{atan2}(-2v + 1, 2u - 1), \phi = \pi\sqrt{(2u - 1)^2 + (2v - 1)^2}\right)$$

$$(D_x, D_y, D_z) = (\sin\phi\cos\theta, \sin\phi\sin\theta, -\cos\phi)$$

The angular map format (Figure 9.18(b)) is similar in appearance to the mirrored sphere mapping, but it samples the directions in a manner that avoids undersampling the regions around the edges. In this mapping, the distance of a point from the center of the image is directly proportional to the angle between straight ahead and its direction in the world. In this way, straight forward appears at the center of the image, and straight right and straight up appear halfway to the edge of the image. Regions that map near the edge of the sphere are sampled with at least as many pixels per degree in any direction as the center of the image. Because of this property, many light probe images (including those in the Light Probe Image Gallery [160]) are available in this format. Unlike the mirrored sphere mapping, the angular map is not equal area, and does not translate solid angle proportionately into image area. Areas near the edge become stretched in the direction tangent to the circumference but are neither stretched nor squeezed in the perpendicular direction, making them overrepresented in the mapping.

Because the mirrored sphere and angular map mappings appear similar, sometimes an angular map image is loaded into a rendering program as if it were a mirrored sphere image, and vice versa. The result is that the environment becomes distorted, and straight vertical lines appear curved. One way to tell which mapping such an image is in is to convert it to the latitude-longitude format (in which vertical lines should be straight) or the cube map format, in which all straight lines should be straight except at face boundaries.

9.4.3 LATITUDE-LONGITUDE

For the latitude-longitude mapping we use a rectangular image domain of $u \in [0, 2]$, $v \in [0, 1]$. The mapping equation for world to image is as follows.

$$(u, v) = \left(1 + \frac{1}{\pi}\operatorname{atan2}(D_x, -D_z), \frac{1}{\pi}\arccos D_y\right)$$

The equation for image to world is as follows.

$$(\theta, \phi) = (\pi(u-1), \pi v)$$
$$(D_x, D_y, D_z) = (\sin\phi\sin\theta, \cos\phi, -\sin\phi\cos\theta)$$

The latitude-longitude mapping (Figure 9.19(a)) maps a direction's azimuth to the horizontal coordinate and its elevation to the vertical coordinate of the image. This is known to cartographers as an *equirectangular mapping*. Unlike the previous two mappings, it flattens the sphere into a rectangular area. The top edge of the image corresponds to straight up, and the bottom edge of the image is straight down. The format most naturally has a 2:1 aspect ratio (360 degrees by 180 degrees), as this introduces the least distortion for regions near the horizon. The areas toward straight

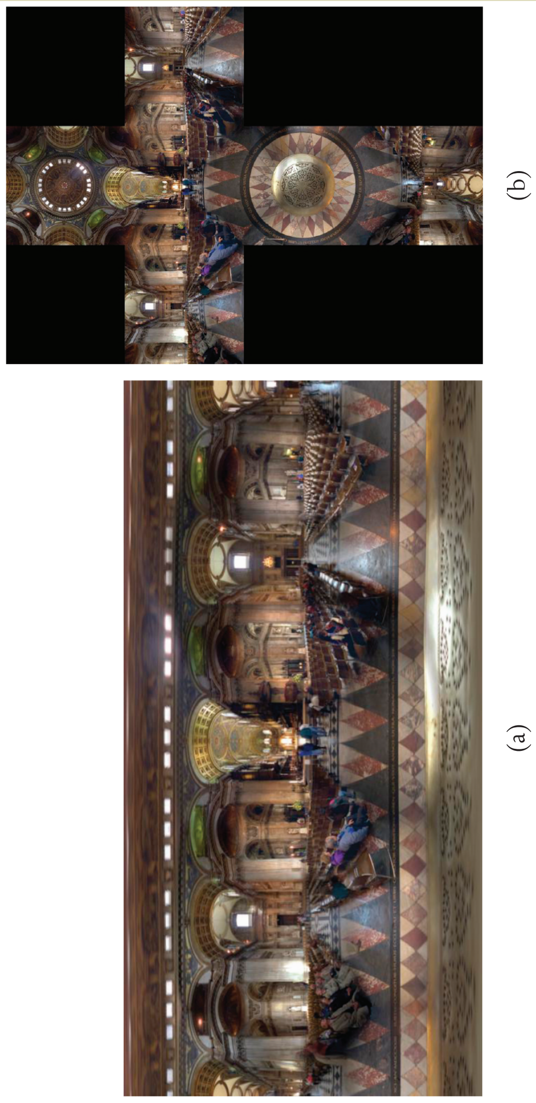


FIGURE 9.19 (a) The latitude-longitude mapping. (b) The cube map mapping.

up and straight down are sampled equivalently with the regions near the horizon in the vertical direction, and are progressively more oversampled toward the top and bottom edges.

The latitude-longitude mapping is convenient because it is rectangular and has no seams (other than at the poles), and because the mapping formulas are simple and intuitive. Although straight lines generally become curved in this format, vertical lines in the scene map to straight vertical lines in the mapping. Another useful property is that in this format the lighting environment can be rotated around the y axis simply by translating the image horizontally. The mapping is not equal area (the percentage any particular area is overrepresented is inversely proportional to the cosine of the latitude ϕ). Thus, to find the average pixel value in a light probe image in this format one can multiply the image by the vertical cosine falloff function $\cos\phi$ and compute the average value of these modified pixels.

9.4.4 CUBE MAP

For the cube map we use a rectangular image domain of $u \in [0, 3]$, $v \in [0, 4]$. The cube map formulas require branching to determine which face of the cube corresponds to each direction vector or image coordinate. Thus, they are presented as pseudocode. The code for world to image is as follows.

```
if ((Dz<0) && (Dz<=-abs(Dx))
           && (Dz<=-abs(Dy))) // forward
  u = 1.5 - 0.5 * Dx / Dz;
  v = 1.5 + 0.5 * Dy / Dz;
else if ((Dz>=0) && (Dz>=abs(Dx))
                 && (Dz>=abs(Dy))) // backward
  u = 1.5 + 0.5 * Dx / Dz;
  v = 3.5 + 0.5 * Dy / Dz;
else if ((Dy<=0) && (Dy<=-abs(Dx))
                 && (Dy<=-abs(Dz)) // down
  u = 1.5 - 0.5 * Dx / Dy;
  v = 2.5 - 0.5 * Dz / Dy;
else if ((Dy>=0) && (Dy>=abs(Dx))
                 && (Dy>=abs(Dz))) // up
  u = 1.5 + 0.5 * Dx / Dy;
  v = 0.5 - 0.5 * Dz / Dy;
else if ((Dx<=0) && (Dx<=-abs(Dy))
                 && (Dx<=-abs(Dz))) // left
  u = 0.5 + 0.5 * Dz / Dx;
  v = 1.5 + 0.5 * Dy / Dx;
else if ((Dx>=0) && (Dx>=abs(Dy))
                 && (Dx>=abs(Dz))) // right
  u = 2.5 + 0.5 * Dz / Dx;
  v = 1.5 - 0.5 * Dy / Dx;
```

The code for image to world is as follows.

```
if u>=1 and u<2 and v<1 // up
  Vx = (u - 1.5) * 2
  Vy = 1.0
  Vz = (v - 0.5) * -2
else if u<1 and v>=1 and v<2 // left
  Vx = -1.0
  Vy = (v - 1.5) * -2
  Vz = (u - 0.5) * -2
else if u>=1 and u<2 and v>=1 and v<2 // forward
  Vx = (u - 1.5) * 2
  Vy = (v - 1.5) * -2
  Vz = -1.0
else if u>=2 and v>=1 and v<2 // right
  Vx = 1.0
  Vy = (v - 1.5) * -2
  Vz = (u - 2.5) * 2
else if u>=1 and u<2 and v>=2 and v<3 // down
  Vx = (u - 1.5) * 2
  Vy = -1.0
  Vz = (v - 2.5) * 2
else if u>=1 and u<2 and v>=3 // backward
  Vx = (u - 1.5) * 2
  Vy = (v - 3.5) * 2
  Vz = 1.0

normalize = 1 / sqrt(Vx * Vx + Vy * Vy + Vz * Vz)

Dx = normalize * Vx
Dy = normalize * Vy
Dz = normalize * Vz
```

In the cube map format (Figure 9.19(b)) the scene is represented as six square perspective views, each with a 90-degree field of view, which is equivalent to projecting the environment onto a cube and then unfolding it. The six squares are most naturally unfolded into a horizontal or vertical cross shape but can also be packed into a 3×2 or 6×1 rectangle to conserve image space. The mapping is not equal area (areas in the corners of the cube faces take up significantly more image area per solid angle than the areas in the center of each face). However, unlike the angular map and latitude-longitude mappings, this relative stretching is bounded: angular areas that map to the cube's corners are overrepresented by a factor of up to $3\sqrt{3}$ in area relative to regions in the center.

This mapping requires six different formulas to convert between world directions and image coordinates, depending on which face of the cube the pixel falls within. Although the equations include branching, they can be more efficient to evaluate than the other mappings (which involve transcendental functions such as *asin* and *atan*2). This image format is sometimes the most convenient for editing the light probe image, because straight lines in the environment remain straight in the image (although there are generally directional discontinuities at face boundaries). For showing a light probe image in the background of a real-time rendering application, the image mapping is straightforward to texture map onto a surrounding cubical surface.

9.5 HOW A GLOBAL ILLUMINATION RENDERER COMPUTES IBL IMAGES

Returning to the rendering process, it is instructive to look at how a global illumination algorithm computes IBL images such as those seen in Figure 9.4. In general, the algorithm needs to estimate how much light arrives from the lighting environment and the rest of the scene at each surface point, which in large part is a matter of visibility: light from the visible parts of the environment must be summed, and light that is blocked by other parts of the scene must instead be replaced by an estimate of the light reflecting from those surfaces. This measure of incident illumination, multiplied by the reflectance of the surface itself, becomes the color rendered at a particular pixel in the image.

The RNL animation was rendered with RADIANCE [205], which like most modern global illumination systems is based on ray tracing. The image is rendered one pixel at a time, and for each pixel the renderer needs to determine the RGB color L to display for that pixel. In our case, L is an HDR RGB pixel value, with its three components proportional to the amount of red, green, and blue radiance arriving toward the camera in the direction corresponding to the pixel. For each pixel, a ray R is traced from the camera C (Figure 9.20(a)) until it hits a surface in the scene at a 3D point P. L is then computed as a function of the reflectance properties of the surface at P and the incident light arriving at P. This section lists the different types of surfaces R can hit and how the rendering system then computes their appearance as lit by the scene. We will see that the most costly part of the process is computing how light reflects from diffuse surfaces. Later in this chapter, we will see how understanding this rendering process can motivate techniques for increasing the rendering efficiency.

Case 1: *R* Hits the IBL Environment If the ray strikes the emissive surface surrounding the scene at point P (Figure 9.20(a)), the pixel color L in the rendering is computed as the color from the light probe image that was texture mapped onto the surface at P. In RNL, this could be a green color from the leaves of the trees, a bright blue color from the sky, or a brown color from the ground below, depending on which pixel and where the camera is looking. The HDR range of the surrounding environment is transferred to the resulting HDR renderings.

In this case, the renderer does not take into consideration how the surface at point P is being illuminated, because the surface is specified to be emissive rather than reflective. This is not exactly how light behaves in the real world, in that most objects placed in a scene have at least a minute effect on the light arriving at all other surfaces in the scene. In RNL, however, the only place where this might be noticeable is on the ground close to the pedestal, which might receive less light

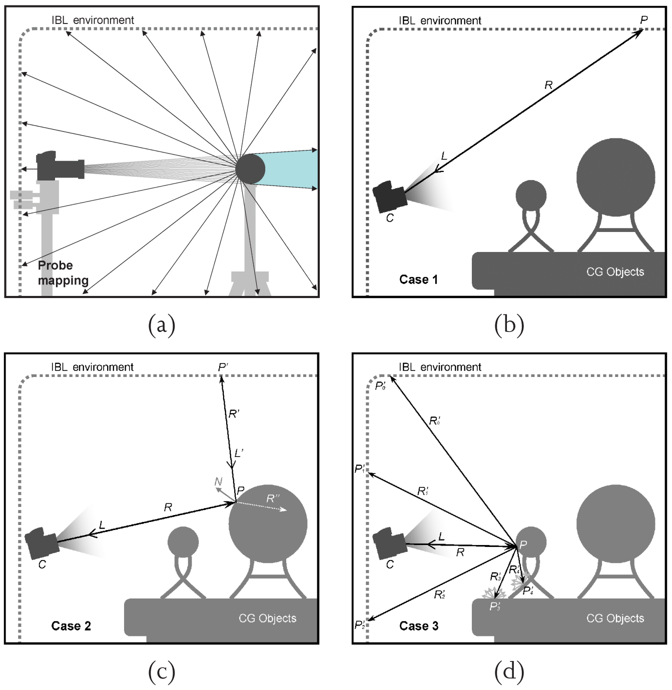


FIGURE 9.20 **Probe mapping:** *A mirrored sphere reflects the entire surrounding environment except for the light blue area obscured by the ball. The ideal mirrored sphere mapping (see Section 9.4.1) maps a light probe image onto an IBL environment surface.* **Case 1:** *A ray R sent from the virtual camera C hits the IBL environment surface at P. The HDR value L on the surface is copied to the image pixel.* **Case 2:** *R hits a specular surface of a CG object. The reflected ray R' is traced, in this case striking the IBL environment at P' with an HDR value of L'. L' is multiplied by the object's specular color to determine the final pixel value L. (For a translucent surface, a refracted ray R'' is also traced.)* **Case 3:** *R hits a diffuse surface at P. A multitude of rays R_i' are traced into the scene to determine the irradiance E at P as a weighted average of the incident light values L_i' from points P_i'. E is multiplied by the diffuse object color to determine the final pixel value L.*

from the environment due to shadowing from the pedestal. Because this area was not seen in the RNL animation, the effect did not need to be simulated.

Case 2: R Hits a Mirror-like Specular Surface A mirror-like specular surface having no roughness shows a clear image of the scene around it. In RNL, the glass, plastic, and metallic spheres have mirror-like specular components. When the ray R hits such a surface at a point P (Figure 9.20(b)), the renderer reflects the ray about P's surface normal N and follows this reflected ray R' into the scene until it strikes a new surface at point P'. It then recursively computes the color L' of the light coming from P' along R' toward P in precisely the same way it computes the light coming from a point P along R toward the camera. For example, if R' strikes the emissive surface surrounding the scene at P', the renderer retrieves the appropriate pixel color L' from the surrounding light probe image (an instance of Case 1). The recursion depth for computing bounces of light is usually limited to a user-defined number, such as six bounces for specular reflections and two bounces for diffuse reflections.

The incident light L' along the reflected ray is then used to produce the specular component of the resulting light L reflecting from the surface. For metals, this light is multiplied by the metallic color of the surface. For example, a gold material might have a metallic color of (0.8, 0.6, 0.3). Because these color components are less than 1, the metallic reflection will reveal some of the detail in the reflected HDR environment not seen directly.

For glass-like materials, the mirror-like specular component is fainter, typically in the range of 4 to 8% (though at grazing angles it becomes greater due to Fresnel reflection), depending on the index of refraction of the material. Thus, the light L' is multiplied by a small value to create the specular component of the color L in the image. In the case of RNL, this makes the bright sky detail particularly evident in the reflections seen in the top of the large glass ball (as in Figure 9.4(a)). For glass-like materials, a second refracted ray R'' is also traced through the translucent surface, and the light L'' arriving along this ray is added to the total light L reflected toward the camera.

For plastic materials, there is both a specular component and a diffuse component to the reflection. For such materials the specular component is computed in the same way as the specular component of glass-like materials and is added to the diffuse component, which is computed as described in material following.

Case 3: R Hits a Diffuse Surface Diffuse surfaces reflect light equally in all directions, and light arriving from every direction in the upper hemisphere of the surface contributes to the reflected light. Because of this, computing the light reflecting from diffuse surfaces can be computationally expensive. The total amount of light arriving at a surface point P is called its *irradiance*, denoted by E, which is a weighted integral of all colors L_i' arriving along all rays R_i' (Figure 9.20(c)). The contribution of each ray R_i' is weighted by the cosine of the angle θ_i between it and the surface normal N, because light arriving from oblique angles provides less illumination per unit area. If we denote $L'(P,\omega)$ to be the function representing the incident light arriving at P from the angular direction ω in P's upper hemisphere Ω, E can be written as

$$E(P,N) = \int_{\Omega} L'(P,\omega)\cos\theta \, d\omega.$$

Unfortunately, E cannot be computed analytically, because it is dependent on not just the point's view of the environment but also on how this light is occluded and reflected by all other surfaces in the scene visible to the point. To estimate E, the renderer takes a weighted average of the light colors L_i' arriving from a multitude of rays R_i' sent out from P to sample the incident illumination. When a ray R_i' strikes the surface surrounding the scene, it adds the corresponding pixel color L_i' from the lighting environment to the sum, weighted by the cosine of the angle between N and R_i'. When a ray R_i' strikes another part of the scene P_i', the renderer recursively computes the color of light L_i' reflected from P_i' toward P and adds this to the sum as well, again weighted by $\cos\theta$. Finally, E is estimated as this sum divided by the total number of rays sent out, as follows.

$$E(P,N) \approx \frac{1}{k}\sum_{i=0}^{k-1} L_i' \cos\theta_i$$

The accuracy of the estimate of E increases with the number of rays k sent out from P. Once E is computed, the final light L drawn for the pixel is the surface's diffuse color (called its *albedo*, often denoted by ρ) multiplied by the irradiance E.

Computing the integral of the incident illumination on an object's surface performs a blurring process on the HDR light probe image that is similar to the blurring we have seen in Section 9.2.5. HDR pixels in the image are averaged together according to a filter, in this case a blur over the upper hemisphere. Just as before, this process makes the effect of HDR pixel values in an environment visible in the much lower dynamic range values reflecting from diffuse surfaces. Clipping the lighting environment to a display's white point before computing the lighting integral would significantly change the values of the computed irradiance.

Because many rays must be traced, the process of sampling the light arriving from the rest of the scene at a point can be computationally intensive. When rays are sent out at random, the number of rays needed to estimate the incident illumination to a given expected accuracy is proportional to the variance of the illumination within the scene. To conserve computation, some renderers (such as RADIANCE) can compute E for a subset of the points P in the scene, and for other points interpolate the irradiance from nearby samples — a process known as *irradiance caching*. Furthermore, irradiance samples computed for one frame of an animation can be reused to render subsequent frames, as long as the scene remains static. Both of these features were used to reduce the rendering time for the RNL animation by a factor of several thousand.

9.5.2 SAMPLING OTHER SURFACE REFLECTANCE TYPES

In RNL, all of the surfaces had either a mirror-like specular component or completely diffuse reflectance, or a combination of the two. Other common material types include rough specular reflections and more general bidirectional reflectance distribution functions (BRDFs) [190] that exhibit behaviors such as retroreflection and anisotropy. For such surfaces, the incident illumination needs to be sampled according to these more general distributions. For example, in the case of rough specular surfaces the rendering system needs to send out a multitude of rays in the general direction of the reflected angle R' with a distribution whose spread varies with the specular roughness parameter. Several BRDF models (such as Lafortune et al. [178] and Ashikhmin and Shirley [153]) have associated sampling algorithms that generate reflected ray directions in a distribution that matches the relative contribution of incident light directions for any particular viewing direction. Having such *importance sampling* functions is very helpful for computing renderings efficiently, as the alternative is to send out significantly more rays with a uniform distribution and weight the incident light arriving along each ray according to the BRDF. When the number of rays is limited, this can lead to noisy renderings in comparison to sampling in a distribution that matches the BRDF. Figure 9.21 shows an example of this difference from Lawrence et al. [181].

In general, using more samples produces higher-quality renderings with more accurate lighting and less visible noise. We have just seen that for different types of materials renderings are created most efficiently when samples are chosen according to a distribution that matches the relative importance of each ray to the final appearance of each pixel. Not surprisingly, the importance of different ray direc-

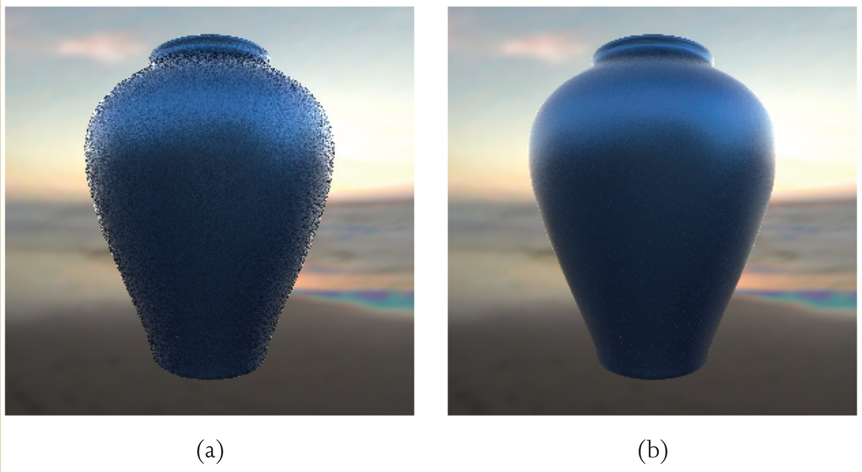


FIGURE 9.21 *IBL renderings of a pot from [181] computed with 75 rays per pixel to sample the incident illumination for a vase with a rough specular BRDF. (a) Noisy results obtained by using uniform sampling and modulating by the BRDF. (b) A result with less noise obtained by sampling with a distribution based on a factored representation of the BRDF.*

tions depends on not only the BRDF of the material but on the distribution of the incident illumination in the environment. To render images as efficiently as possible, it becomes important to account for the distribution of the incident illumination in the sampling process, particularly for IBL environments with bright concentrated light sources. The next section presents this problem and describes solutions for sampling incident illumination efficiently.

9.6 SAMPLING INCIDENT ILLUMINATION EFFICIENTLY

We have just seen that the speed at which images can be computed using IBL as described in the previous section depends on the number of rays that need to be traced. The bulk of these rays are traced to estimate the illumination falling upon diffuse surfaces through sampling. The light falling on a surface (i.e., its irradiance E) is the average value of the radiance arriving along all light rays striking the surface, weighted by the cosine of the angle each ray makes with the surface normal. Because averaging together the light from every possible ray is impractical, global illumination algorithms estimate the average color using just a finite sampling of rays. This works very well when the lighting environment is generally uniform in color. In this case the average radiance from any small set of rays will be close to the average from all of the rays because no particular ray strays far from the average value to begin with. However, when lighting environments have some directions that are much brighter than the average color, it is possible for the average of just a sampling of the rays to differ greatly from the true average.

When ray directions are chosen at random, the number of rays needed to accurately sample the illumination is proportional to the variance in the light probe image. Lighting environments that have low variance (such as cloudy skies) can be sampled accurately with just tens of rays per irradiance calculation. Environments with greater variance, such as scenes with concentrated light sources, can require hundreds or thousands of rays per irradiance calculation when rays are sent out at random. The Eucalyptus Grove light probe, which featured bright backlit clouds in the direction of the setting sun, required over 1,000 rays per irradiance calculation, making rendering RNL computationally expensive. In this section, we describe this sampling problem in detail and present several sampling techniques that mitigate the difficulties.

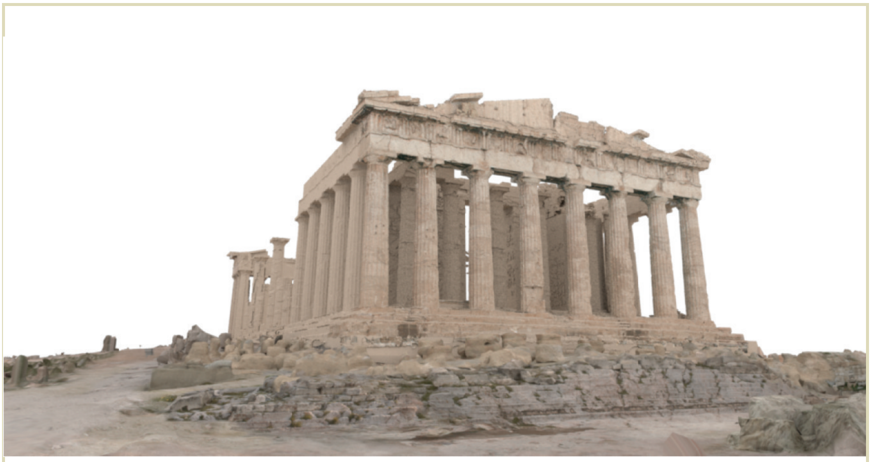


FIGURE 9.22 *A laser-scanned 3D model of the Parthenon illuminated by a uniformly white lighting environment, giving the appearance of a dense cloudy day.*

Figure 9.22 shows a virtual model of the Parthenon rendered with IBL using a synthetic lighting environment of a completely white sky.[5] This environment, being all the same color, has zero variance, and a high-quality rendering could be computed using the Arnold global illumination rendering system [167] using a relatively modest 81 rays to sample the incident light at each pixel. The only source of illumination variance at each surface point is due to visibility: some directions see out to the sky, whereas others see indirect light from the other surfaces in the scene. Because the color of the indirect light from these other surfaces is also low in variance and not radically different from the light arriving from the sky, the

5 The virtual Parthenon model was created by laser scanning the monument and using an inverse global illumination process leveraging image-based lighting to solve for its surface colors and reflectance properties from photographs, as described in Debevec et al. [163].

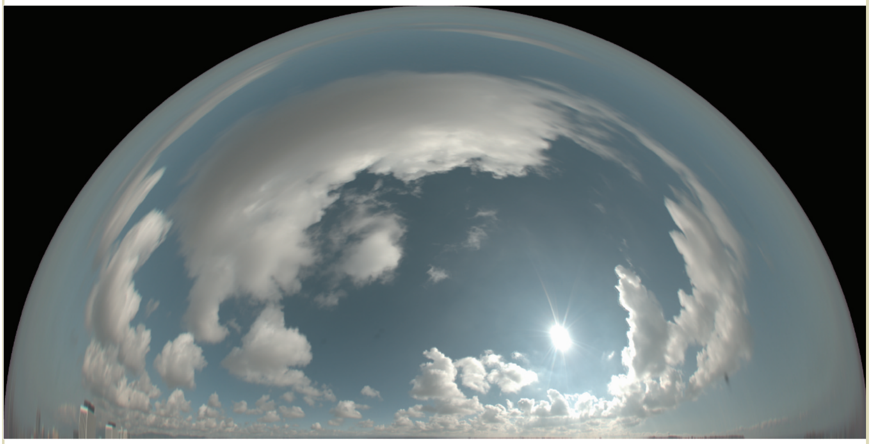


FIGURE 9.23 *A light probe image of a sunny sky, taken with a fish-eye lens. Pixels on the sun disk (much smaller than the saturated region seen here) are on the order of 100,000 times brighter than the average color of the rest of the sky.*

total variance remains modest and the illumination is accurately sampled using a relatively small number of rays.

In contrast to the perfectly even sky, Figure 9.23 shows a clear sky with a directly visible sun, acquired using the technique in Stumpfel et al. [200]. The half-degree disk of the sun contains over half the sky's illumination but takes up only one hundred thousandth of the sky's area, giving the sunny sky a very high variance. If we use such a high-variance environment to illuminate the Parthenon model, we obtain the noisy rendering seen in Figure 9.24. For most of the pixels, none of the 81 rays sent out to sample the illumination hit the small disk of the sun, and thus they appear only to be lit by the light from the sky and clouds. For the pixels where one of the rays did hit the sun, the sun's contribution is greatly overestimated, because the sun is counted as $\frac{1}{81}$ of the lighting environment rather than one hundred

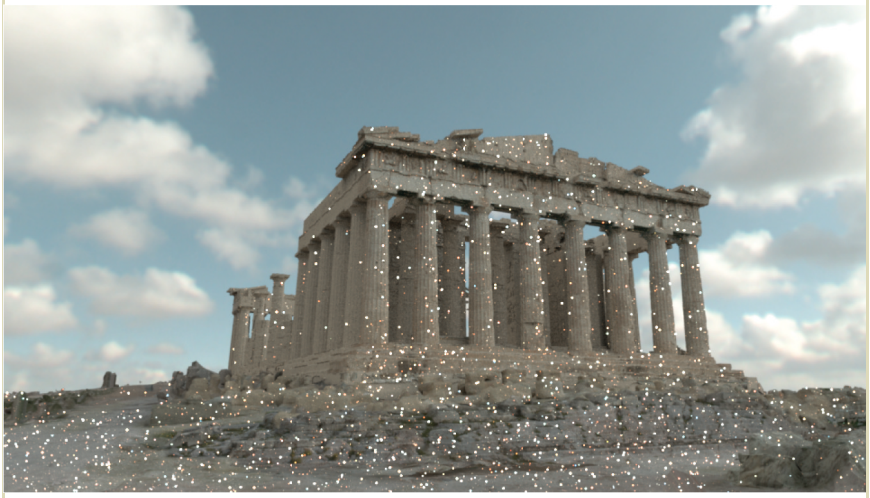


FIGURE 9.24 *The virtual Parthenon rendered using the sunny sky light probe in Figure 9.23 with 81 lighting samples per pixel. The appearance is speckled because 81 samples distributed randomly throughout the sky are not enough to accurately sample the small disk of the sun.*

thousandth. These pixels are far brighter than what can be shown in an LDR image, and their intensity is better revealed by blurring the image somewhat (as in Figure 9.25). On average, this image shows the correct lighting, but the appearance is strange because the light from the sun has been squeezed into a random scattering of overly bright pixels.

The noisy Parthenon rendering shows what is known as the *sampling problem*. For lighting environments with concentrated sources, a very large number of rays needs to be sent in order to avoid noisy renderings. For the sunny sky, hundreds of thousands of rays would need to be sent to reliably sample the small sun and the large sky, which is computationally impractical. Fortunately, the number of rays can be

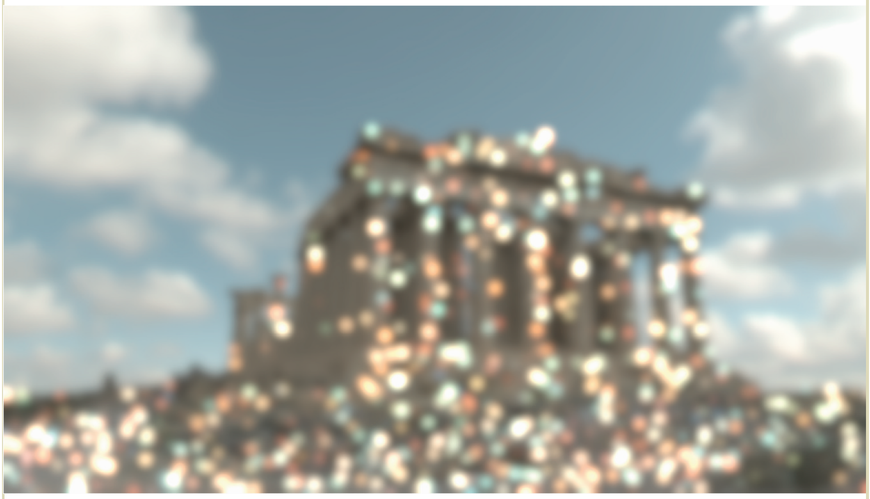


FIGURE 9.25 *A Gaussian blurred version of Figure 9.24, showing more clearly the amount of reflected sun energy contained in just a few of the image pixels.*

reduced to a manageable number using several techniques. In each technique, the key idea is to give the rendering algorithm *a priori* information about how to sample the illumination environment efficiently. These techniques (described in material following) include *light source identification*, *light source constellations*, and *importance sampling*.

9.6.1 IDENTIFYING LIGHT SOURCES

The ray sampling machinery implemented in traditional global illumination algorithms typically only samples the indirect lighting within a scene. Concentrated light sources are assumed to be explicitly modeled and taken into account in the direct lighting calculation in which rays are sent to these lights explicitly. In image-based lighting, both direct and indirect sources are effectively treated as indirect illumination, which strains the effectiveness of this sampling machinery.

Many IBL environments can be partitioned into two types of areas: small concentrated light sources and large areas of indirect illumination. These types of environments fare poorly with simplistic sampling algorithms, in that much of the illumination is concentrated in small regions that are easily missed by randomly sampled rays. One method of avoiding this problem is to identify these small concentrated light regions and convert them into traditional area light sources. These new area light sources should have the same shape, direction, color, and intensity as seen in the original image, and the corresponding bright regions in the image-based lighting environment must be removed from consideration in the sampled lighting computation. This yields the type of scene that traditional global illumination algorithms are designed to sample effectively.

For the sunny sky light probe, this process is straightforward. The direction of the sun can be determined as the centroid of the brightest pixels in the image, converted to a world direction vector using the angular mapping formula in Section 9.4.2. As mentioned earlier in this chapter, the size and shape of the sun is known to be a disk whose diameter subtends 0.53 degrees of the sky. The color and intensity of the sun can be obtained from the light probe image as the average RGB pixel value of the region covered by the sun disk. In a typical rendering system a specification for such a light source might look as follows.

```
light sun directional {
  direction -0.711743 -0.580805 -0.395078
  angle 0.532300
  color 10960000 10280000 866000
}
```

Figure 9.26 shows a global illumination rendering of the Parthenon illuminated just by the sun light source. For each pixel, the renderer explicitly sends at least one ray toward the disk of the sun to sample the sun's light because the sun is a direct light source known *a priori* to the renderer. Thus, the rendering has no noise problems, as in Figure 9.24. Although the rest of the sky is black, the renderer still sends additional randomly fired rays from each surface to estimate the indirect illumination arriving from other surfaces in the scene. These effects are most significant in the case of shadowed surfaces that are visible to sunlit surfaces, such as the left sides of

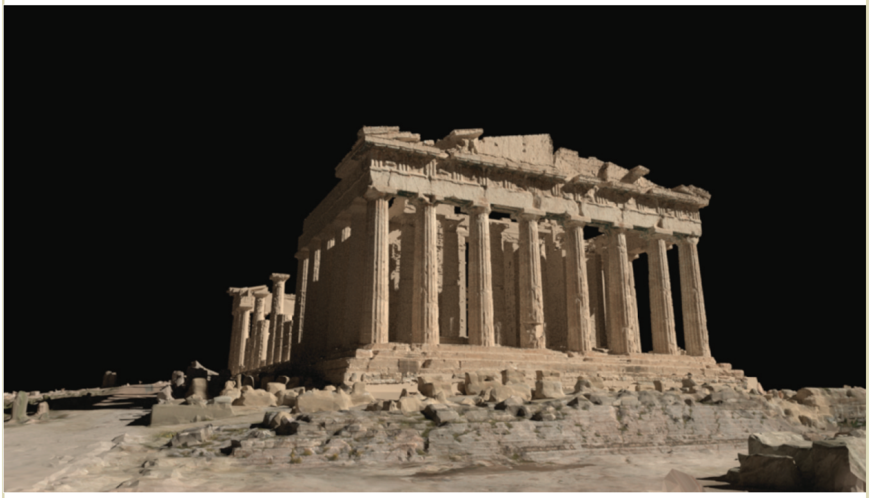


FIGURE 9.26 *The Parthenon illuminated only by the sun, simulated as a direct light source.*

the front columns. Because the blue skylight scattered by the atmosphere is not being simulated, the rendering shows how the Parthenon might look if it were located on the moon and illuminated by a somewhat yellowish sun.

The rest of the sky's illumination can be simulated using an image-based lighting process, but we first need to make sure that the light from the sun is no longer considered to be part of the image-based lighting environment. In some rendering systems, it is sufficient to place the new direct light source in front of the IBL environment surface and it will occlude the image-based version of the source from being hit by indirect sample rays. In others, the corresponding image region should be set to black in order to prevent it from being part of the image-based illumination. If we remove the sun from the sky and use the remainder of the image as an IBL environment, we obtain the rendering seen in Figure 9.27. This rendering,

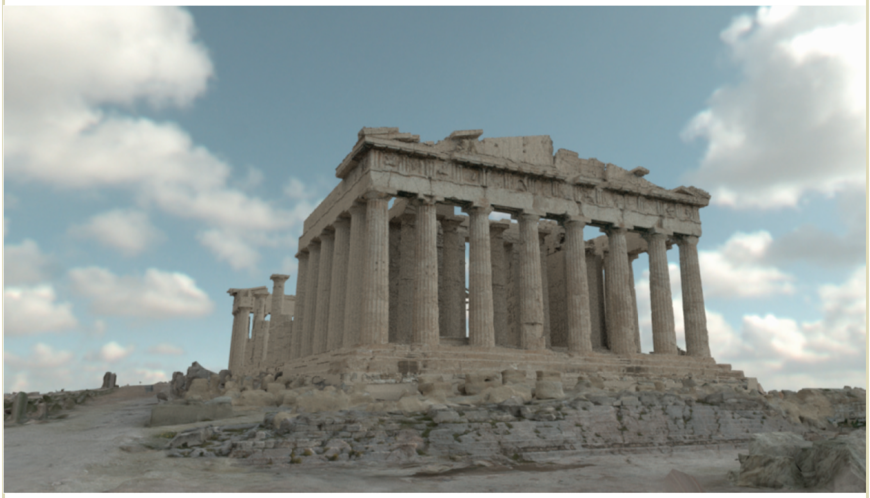


FIGURE 9.27 *The Parthenon illuminated only by the sky and clouds, with 81 samples per pixel.*

although it lacks sunlight, is still a realistic one. It shows the scene approximately as if a cloud had passed in front of the sun.

In practice, it is sometimes necessary to delete a somewhat larger area around the light source from the IBL environment, because some light sources have significantly bright regions immediately near them. These regions on their own can contribute to the appearance of noise in renderings. Sometimes, these regions are due to glare effects from imperfect camera optics. In the case of the sun, forward scattering effects in the atmosphere creates a bright circumsolar region around the sun. Because of both of these effects, to create the sky used for Figure 9.27 a region covering the circumsolar area was removed from the light probe image, and this additional light energy was added to the sun intensity used to render Figure 9.26.

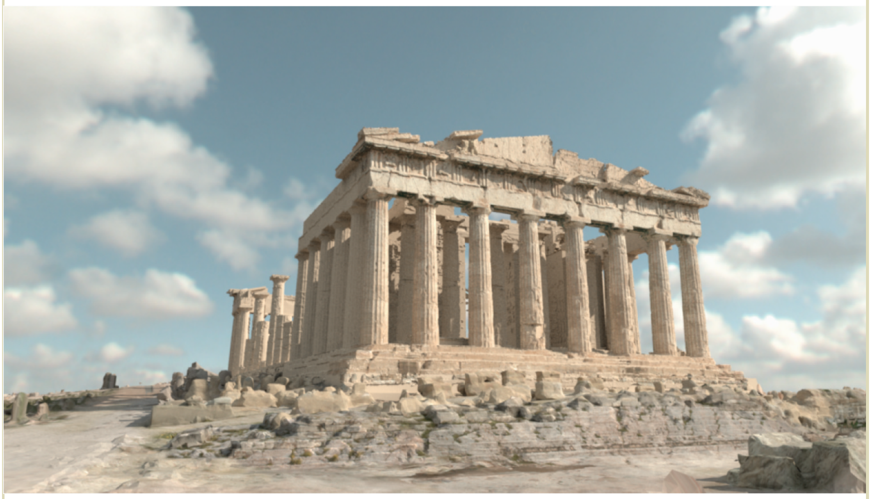


FIGURE 9.28 *An efficient noise-free rendering of the Parthenon made using IBL for the sky and clouds and a direct light source for the sun, with 81 samples per pixel.*

As a result, the edges of shadows in the sun rendering are slightly sharper than they should be, but the effect is a very subtle one.

Finally, we can light the scene with the sun and sky by simultaneously including the IBL environment for the sky and clouds and the direct light source for the sun in the same rendering, as seen in Figure 9.28. (We could also just add the images from Figures 9.26 and 9.27 together.) This final rendering shows the combined illumination from the sun and the rest of the sky, computed with a relatively modest 81 sample rays per pixel.

Identifying light sources can also be performed for more complex lighting environments. The SIGGRAPH 99 Electronic Theater animation *Fiat Lux* used an IBL environment created from HDR images acquired in St. Peter's Basilica. It was rendered with the RADIANCE lighting simulation system. The images were assembled into HDR panoramas and projected onto the walls of a basic 3D model of the Basilica's interior, seen in Figure 9.29(b). The illumination within the Basilica consisted of indirect light from the walls and ceiling, as well as concentrated illumination from the windows and the incandescent lights in the vaulting. To create renderings efficiently, each of the concentrated area lights' sources was identified by drawing a polygon around the source in the panoramic image, as seen in Figure 9.29(a). A simple program computed the average HDR pixel value of the region covered by each light source, and created an area light source set to this value for its radiance. Just like the panoramic image, the vertices of the identified light sources were projected onto the walls of the virtual Basilica model, placing the lights at their proper 3D locations within the scene. Thus, the image-based illumination changed throughout the virtual scene, as the directional light depended on each object's 3D location relative to the light sources.

Two other details of the procedure used in *Fiat Lux* are worth mentioning. First, the light sources were placed a small distance in front of the walls of the Basilica, so that rays fired out from the surfaces would have no chance of hitting the bright regions behind the lights without having to set these regions to black. Second, the lights were specified to be "illum" light sources, a special RADIANCE light source type that is invisible to rays coming directly from the camera or from mirror-like reflections. As a result, the lights and windows appeared with their proper image-based detail when viewed directly and when seen in the reflections of the synthetic objects, even though they had been covered up by direct light sources. Figure 9.30(a) shows a frame from *Fiat Lux* in which a variety of virtual objects have been placed within the Basilica using IBL with identified light sources. Because "illum" sources were used, the motion-blurred reflections in Figure 9.30(b) reflect the original image-based light sources.

Identifying light sources dramatically reduces the number of rays needed to create noise-free renderings of a scene, and it maintains the realism and accuracy of image-based lighting. Having the concentrated sources converted to individual CG lights is also useful for art direction, as these light sources can be readily repositioned and changed in their color and brightness. These light sources can also be used in a rendering system that does not support global illumination, yielding at least an approximate version of the IBL environment. For some applications, however, it is desirable to have a fully automatic method of processing an IBL en-

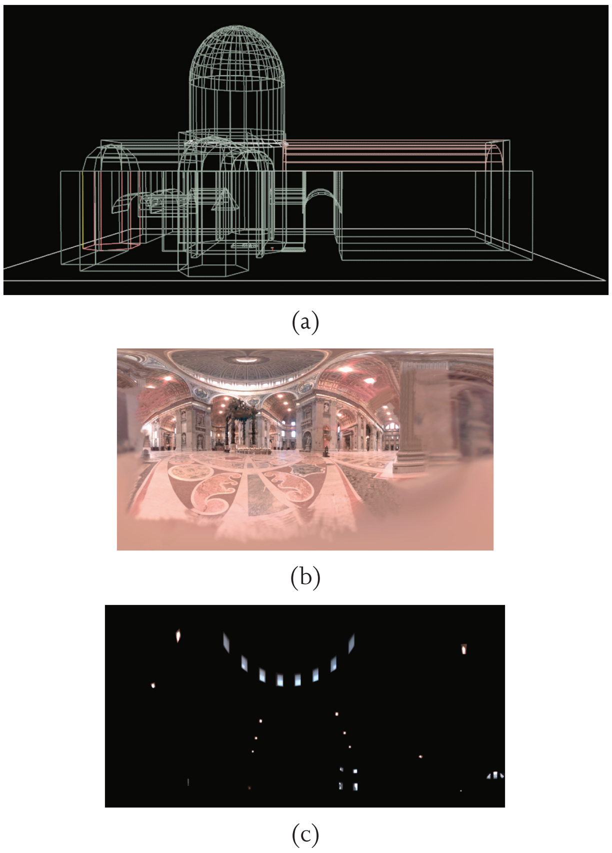
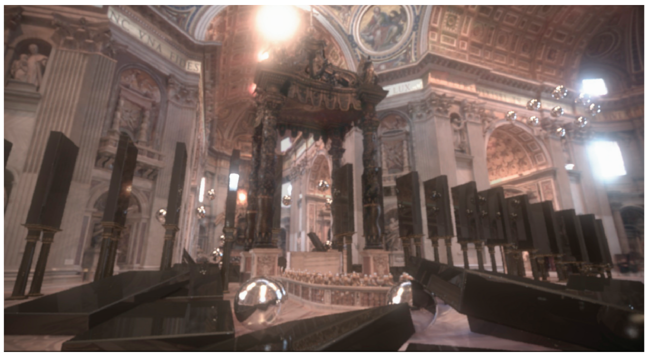
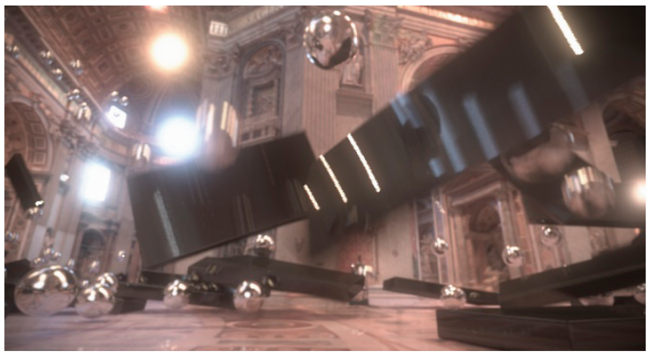


FIGURE 9.29 (a) Identified light sources from (c) corresponding to the windows and the incandescent lights in the vaulting. The HDR image and the light sources were projected onto a 3D model of the Basilica interior to form a 3D lighting environment. (b) Basic 3D geometry of the Basilica interior used as the IBL environment surface. (c) An image-based lighting environment acquired inside St. Peter's Basilica.



(a)



(b)

FIGURE 9.30 (*a*) *A frame from the Fiat Lux animation, showing synthetic objects inserted into the Basilica, illuminated by the HDR lighting environment using identified light sources.* (*b*) *Another frame from the animation showing HDR motion blur effects in the reflections of the spinning objects. Shadows and reflections of the objects in the floor were created using the techniques described in Section 9.7.*

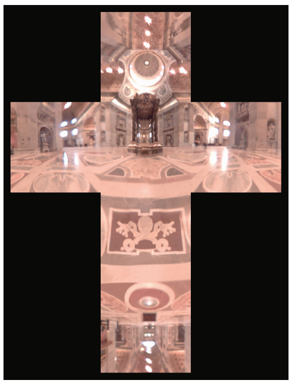
vironment into a description that can be rendered efficiently. The remainder of this section presents some of these automatic sampling techniques.

9.6.2 CONVERTING A LIGHT PROBE INTO A CONSTELLATION OF LIGHT SOURCES

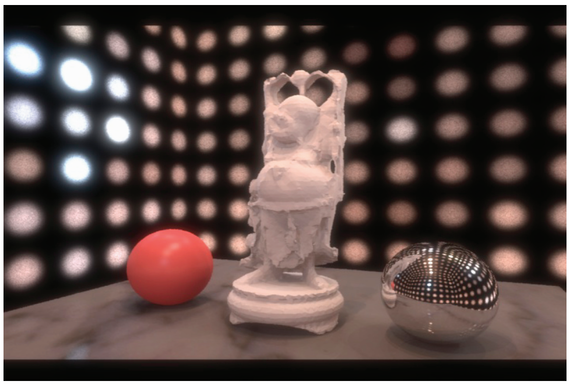
As we saw previously, it is possible to reduce the variance in an image-based lighting environment by converting concentrated spots of illumination into direct light sources. We can carry this idea further by turning *entire* lighting environments into constellations of light sources. These approximations can eliminate noise from renderings but can introduce aliasing in shadows and highlights when not enough light sources are used. When the light source directions are chosen with care, accurate illumination can be created using a manageable number of light sources, and these lights can be used in either traditional or global illumination rendering systems.

In general, this approach involves dividing a light probe image into a number of regions, and then creating a light source corresponding to the direction, size, color, and intensity of the total light coming from each region. Each region can be represented either by a point light source, or by an area light source corresponding to the size and/or shape of the region. Figure 9.31 shows perhaps the case of approximating an IBL environment with a constellation of point light sources. A light probe taken within St. Peter's Basilica was converted to a cube map (Figure 9.31(a)), and each face of the cube map was resized to become a square of just 10 by 10 pixels, seen in Figure 9.31(a). Then, a point light source was placed in the direction corresponding to each pixel on each face of the cube, and set to the color and intensity of the corresponding pixel, yielding 600 light sources. These light sources produced a low-resolution point-sampled version of the image-based lighting environment. The technique has the attractive quality that there is no sampling noise in the renderings, as rays are always traced to the same light locations. However, the technique can introduce *aliasing*, because the finite number of lights may become visible as stair-stepped shadows and fragmented specular reflections.

Figure 9.31(b) shows the results of rendering a small scene using this constellation of light sources. For both the diffuse figure and the glossy red ball the rendering is free of noise and artifacts, though the shadows and highlights from the windows and lights are not in precisely the right locations due to the finite resolution of the



(a)



(b)

FIGURE 9.31 *(a) A light probe image taken inside St. Peter's Basilica in cube map format. (b) An approximation of the St. Peter's lighting using a* 10×10 *array of point lights for each cube face. The Buddha model is courtesy of the Stanford computer graphics laboratory.*

light source constellation. For the shiny sphere, simulating the illumination from the set of point lights makes little sense because there is a vanishingly small probability that any particular reflected ray would precisely hit one of the point lights. Instead, it makes sense to use ray tracing to reflect these rays directly into the image-based lighting environment, as described in case 2 in Section 9.5. In the rendering, the mirrored sphere is shown reflecting an illustrative image composed of spots for each point light source.

The quality of approximating an IBL environment with a finite number of lights can be significantly increased if the light sources are chosen in a manner that conforms to the distribution of the illumination within the light probe image. One strategy for this is to have each light source represent approximately the same quantity of light in the image. Taking inspiration from Heckbert's median-cut color quantization algorithm [173], we can partition a light probe image in the rectangular latitude-longitude format into 2^n regions of similar light energy as follows.

1. Add the entire light probe image to the region list as a single region.
2. For each region, subdivide along the longest dimension such that its light energy is divided evenly.
3. If the number of iterations is less than n, return to step 2.

For efficiency, calculating the total energy within regions of the image can be accelerated using summed area tables [159]. Once the regions are selected, a light source can be placed in the center of each region, or alternately at each region's energy centroid, to better approximate the spatial distribution of the light within the region. Figure 9.32 shows the Grace Cathedral lighting environment partitioned into 256 light sources, and Figure 9.33 shows a small scene rendered with 16, 64, and 256 light sources chosen in this manner. Applying this technique to our simple diffuse scene, 64 lights produce a close approximation to a well-sampled and computationally intensive global illumination solution, and the 256-light approximation is nearly indistinguishable.

A few implementation details should be mentioned. First, computing the total light energy is most naturally performed on monochrome pixel values rather than RGB colors. Such an image can be formed by adding together the color channels of the light probe image, optionally weighting them in relation to the human eye's sen-

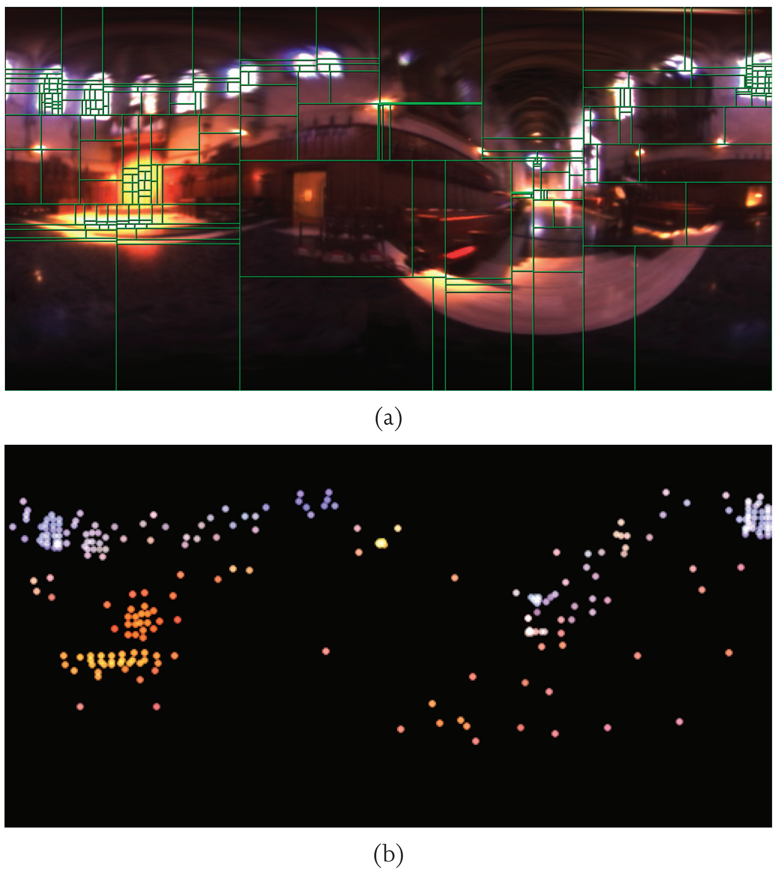


FIGURE 9.32 (a) The Grace Cathedral light probe image subdivided into 256 regions of equal light energy using the median cut algorithm. (b) The 256 light sources chosen as the energy centroids of each region. All of the lights are approximately equal in intensity. A rendering made using these lights appears in Figure 9.33(c).

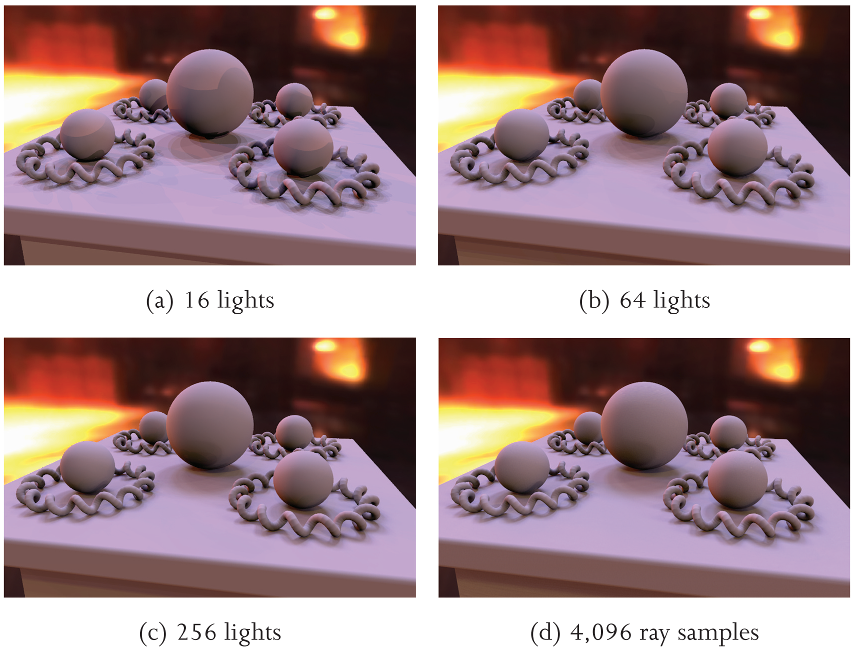


FIGURE 9.33 *Noise-free renderings (a through c) in the Grace Cathedral lighting environment as approximated by 16, 64, and 256 light sources chosen with the median cut algorithm. An almost noise-free Monte Carlo IBL rendering (d) needing 4,096 randomly chosen rays per pixel.*

sitivity to each color channel.[6] Partitioning decisions are made on this monochrome image, and light source colors are computed using the corresponding regions in the

6 Following ITU-R BT.709, the formula used to convert RGB color to monochrome luminance is $Y = 0.2125R + 0.7154G + 0.0721B$.

original color image. Second, the latitude-longitude format overrepresents the area of regions near the poles. To compensate for this, the pixels of the probe image should first be scaled by $\cos\phi$. Additionally, determining the longest dimension of a region should also take this stretching into account. This can be approximated by multiplying a region's width by $\cos\phi$, where ϕ is taken to be the angle of inclination of the middle of the region.

Several other light source selection procedures have been proposed for approximating light probe images as constellations of light sources. In each, choosing these point light source positions is also done with a clustering algorithm.

The LightGen plug-in [158] for HDR Shop [202] takes a light probe image in latitude-longitude format and outputs a user-specified number of point light sources in a variety of 3D modeling program formats. LightGen chooses its light source positions using a K-means clustering process [184]. In this process, K light source positions are initially chosen at random. Then the pixels in the light probe image are partitioned into sets according to which light source they are closest to. For each set of pixels, the mass centroid of the set is determined such that a pixel's mass is proportional to its total intensity. Then, each of the K light sources is moved to the mass centroid of its set. The pixels are repartitioned according to the new light source directions and the process is repeated until convergence. Finally, each light source is assigned the total energy of all pixels within its set. Figure 9.34 shows results for $K = 40$ and $K = 100$ light sources for a kitchen light probe image.

LightGen tends to cluster the light sources around bright regions, but these light sources generally contain more energy than the light sources placed in dimmer regions. As a result, dimmer regions receive more samples than they do in the median cut algorithm, but more lights may be necessary to approximate the structure of shadows. For example, if the kitchen window is approximated as six bright point light sources it can be possible to observe multiple distinct shadows from the six sources rather than the expected soft shadow from an area light source.

Kollig and Keller [177] propose several improvements to LightGen's clustering technique. They begin the K-means procedure using a single randomly placed light source and then add in one more random light at a time, each time iterating the K-means clustering procedure until convergence. This process requires additional computation but performs better at placing the light sources within concentrated areas of illumination. They also discuss several procedures of reducing aliasing once

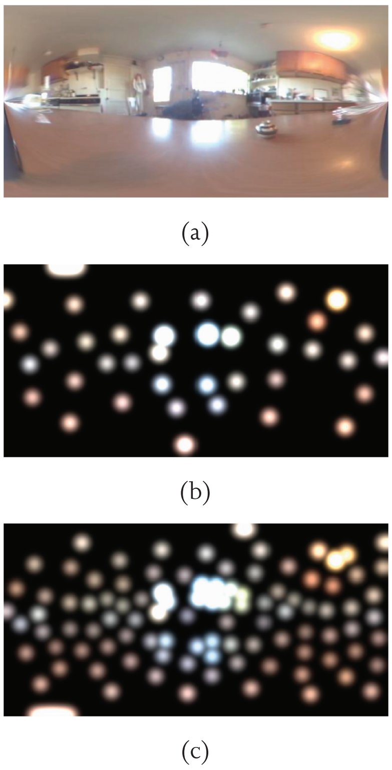


FIGURE 9.34 *(a) An IBL environment of a kitchen. (b) An approximation of the kitchen environment made by LightGen using 40 point lights. (c) An approximation using 100 point lights.*

the light source positions are chosen. One of them is to use the light source regions as a structure for *stratified sampling*. In this process, the lighting environment is sampled with K rays, with one ray chosen to fire at random into each light source region. To

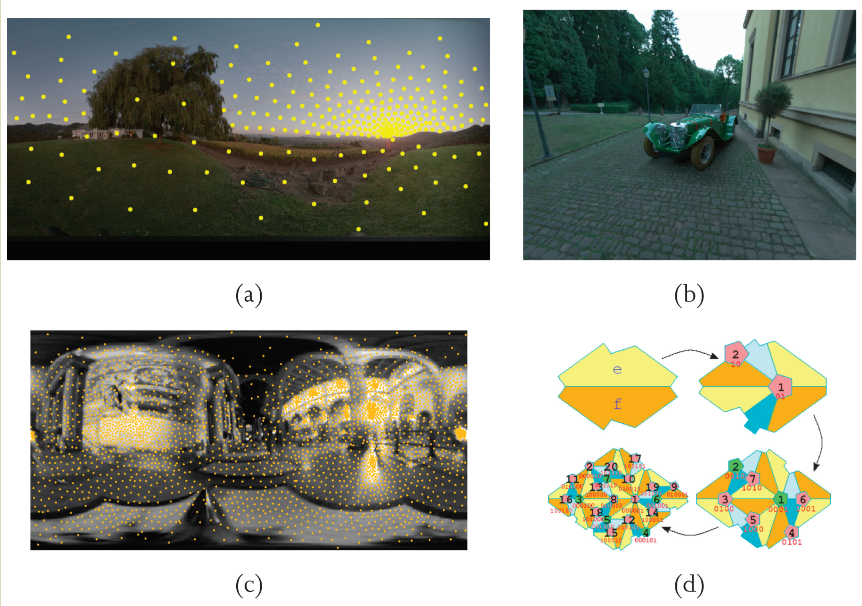


FIGURE 9.35 *(a) Lights chosen for an outdoor light probe image by the Kollig and Keller sampling algorithm. (b) An IBL rendering created using lights from the Kollig and Keller algorithm. (c) Lights chosen for the Galileo's Tomb light probe image by the Ostromoukhov et al. sampling algorithm. (d) The Penrose tiling pattern used by Ostromoukhov et al. for choosing light source positions.*

avoid problems with the remaining variance within each region, they propose using the average RGB color of each region as the color of any ray fired into the region. Lights chosen by this algorithm and a rendering using such lights are shown in Figures 9.35(a) and 9.35(b).

Ostromoukhov et al. [191] use a light source sampling pattern based on the geometric Penrose tiling to quickly generate a hierarchical sampling pattern with an appropriately spaced distribution. The technique is optimized for efficient sampling and was able to achieve sampling speeds of just a few milliseconds to generate several hundred light samples representing a lighting environment. A set of lights chosen using this pattern is shown in Figure 9.35(c), and the Penrose tiling pattern used is shown in Figure 9.35(d).

9.6.3 IMPORTANCE SAMPLING

As an alternative to converting a light probe image into light sources, one can construct a randomized sampling technique such that the rays are sent in a distribution that matches the distribution of energy in the light probe image. In the case of the sunny sky in Figure 9.23, such an algorithm would send rays toward the disk of the sun over half the time, since more than half of the light comes from the sun, rather than in proportion to the sun's tiny area within the image. This form of sampling can be performed using a mathematical technique known as *importance sampling*.

Importance sampling was introduced for the purpose of efficiently evaluating integrals using Monte Carlo techniques [185] and has since been used in a variety of ways to increase the efficiency of global illumination rendering (e.g., Veach and Guibas [204]). Because a computer's random number generator produces uniformly distributed samples, a process is needed to transform uniformly distributed numbers to follow the *probability distribution function* (PDF) corresponding to the distribution of light within the image. The process is most easily described in the context of a 1D function $f(x) : x \in [a, b]$, as seen in Figure 9.36(a). Based on a desired PDF, one computes the cumulative distribution function $g(x) = \int_a^x f(x) / \int_a^b f(x)$, as seen in Figure 9.36(b). We note that $g(x)$ increases monotonically (and thus has an inverse) because $f(x)$ is nonnegative, and that $g(x)$ ranges between 0 and 1. Using $g(x)$, we can choose samples x_i in a manner corresponding to the distribution of energy in $f(x)$ by choosing values y_i uniformly distributed in $[0, 1]$ and letting $x_i = g^{-1}(y_i)$. This process is shown graphically for four samples in Figure 9.36(b).

To see why this process works, note that each bright light source near a point x in the PDF produces a quick vertical jump in the CDF graph in the area of $g(x)$. Seen from the side, the jump produces a flat span whose length is proportional to

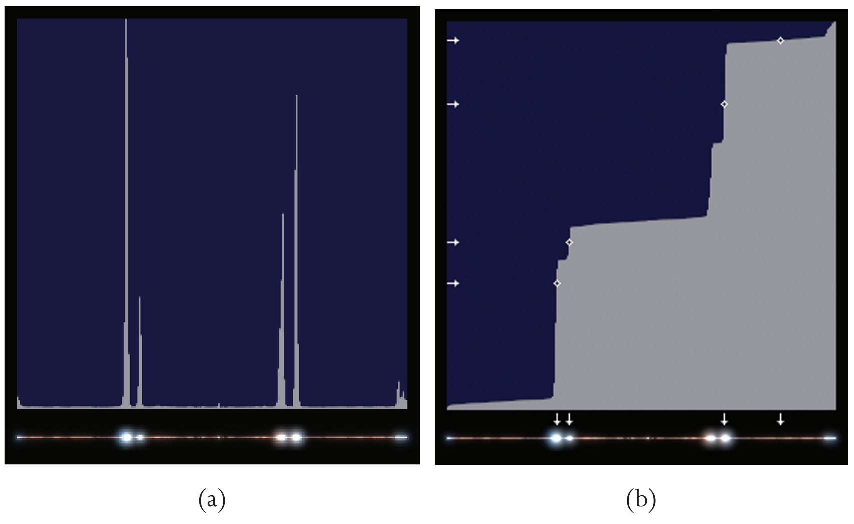


FIGURE 9.36 *Importance sampling. (a) A plot of the brightness of a 256 × 1 pixel region (seen below) of the St. Peter's light probe image that intersects several bright windows, forming a probability distribution function (PDF). The image region is displayed below the graph using an HDR glare effect. (b) A graph of the cumulative distribution function (CDF) of the region from left to right for importance sampling. Four randomly chosen samples y_i (indicated by small horizontal arrows) are followed right until they intersect the graph (indicated by the diamonds), and are then followed down to their corresponding image pixel samples x_i.*

the amount of the scene's energy contained by the light. This flat span becomes a likely landing spot for randomly chosen samples y_i, with a likelihood proportional to the light's energy within the scene. When a sample y_i falls within the span, it produces a sample of the light source near x. In Figure 9.36(b) we see that three of the four random samples fell on spans corresponding to areas within bright light

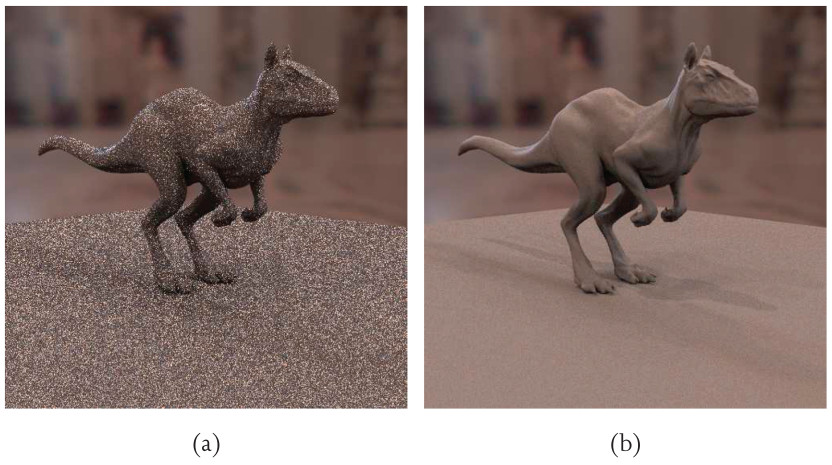


FIGURE 9.37 *(a) A noisy IBL rendering in the St. Peter's lighting environment Figure 9.31(a) using 64 randomly distributed samples per pixel. (b) A much smoother rendering using 64 samples per pixel chosen using the importance sampling technique described in Pharr et al. [192].*

sources. The sample furthest to the top right did not land on a steep slope, and thus produced a sample in one of the dimmer areas of the image.

Extending this technique from 1D functions to 2D images is straightforward, as one can concatenate each row of an $m \times n$ light probe image into a single $mn \times 1$ pixel vector, suggested by Agarwal et al. [152]. Pharr and Humphreys [192] propose an implementation of importance sampling for light probe images within their Physically-Based Rendering package [193], in which importance sampling is first used to compute which column of the image to sample (the energy of each column x is summed to produce $f(x)$) and a sample is taken from the chosen image column using 1D importance sampling (as described previously). Figure 9.37 compares the results of using this sampling algorithm to using randomly distributed rays. As desired, the rendering using importance sampling yields dramatically less noise than random sampling for the same number of samples.

To produce renderings that converge to the correct solution, light values chosen using importance sampling must be weighted in a manner that is inversely proportional to the degree of preference given to them. For example, sampling the sunny sky of Figure 9.23 with importance sampling would on average send over half of the rays toward the sun, as the sun contains over half the light in the image. If we weighted the radiance arriving from all sampled rays equally the surface point would be illuminated as if the sun were the size of half the entire sky.

A deficiency of these light probe sampling techniques is that they sample the lighting environment without regard to which parts of the environment are visible to a given surface point. Agarwal et al. [152] present a *structured importance sampling* approach for IBL that combines importance sampling with the conversion to light sources in order to better anticipate variable visibility to the environment. They first threshhold the image into regions of similar intensity, and use the Hochbaum–Shmoys clustering algorithm to subdivide each region a number of times proportional to both its size and its total energy. In this way, large dim regions do not become undersampled, which improves the rendering quality of shadowed areas. Likewise, small bright regions are represented with fewer samples relative to their intensity, in that their small spatial extent allows them to be accurately modeled with a relatively small number of samples. Cohen [157] constructs a piecewise-constant importance function for a light probe image, and introduces a *visibility cache* that exploits coherence in which parts of the lighting environment are visible to neighboring pixels. With this cache, significantly fewer rays are traced in directions that are occluded by the environment.

9.7 SIMULATING SHADOWS AND SCENE-OBJECT INTERREFLECTION

So far, the IBL techniques presented simulate how the light from an environment illuminates CG objects, but not how the objects in turn affect the appearance of the environment. The most notable effect to be simulated is the shadow an object casts beneath itself. However, shadows are just part of the lighting interaction that should be simulated. Real objects also reflect light back onto the scene around them.

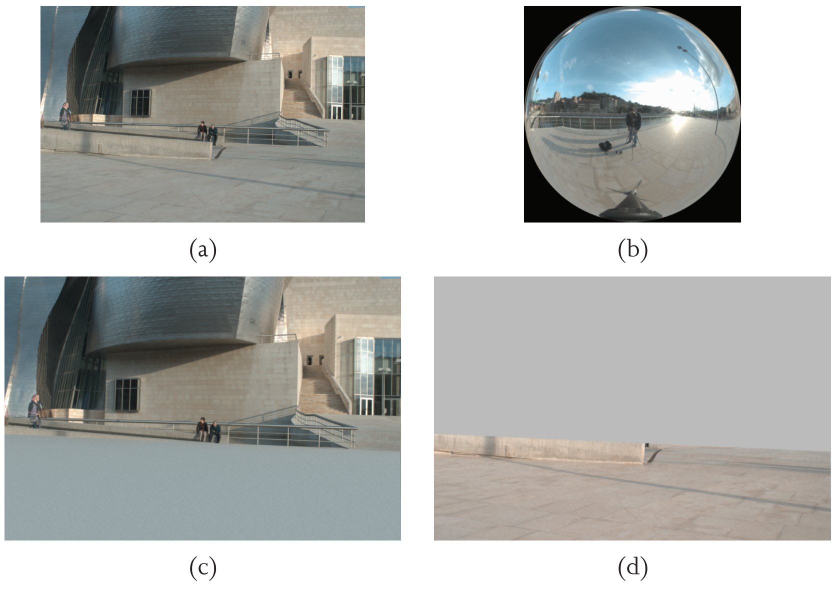
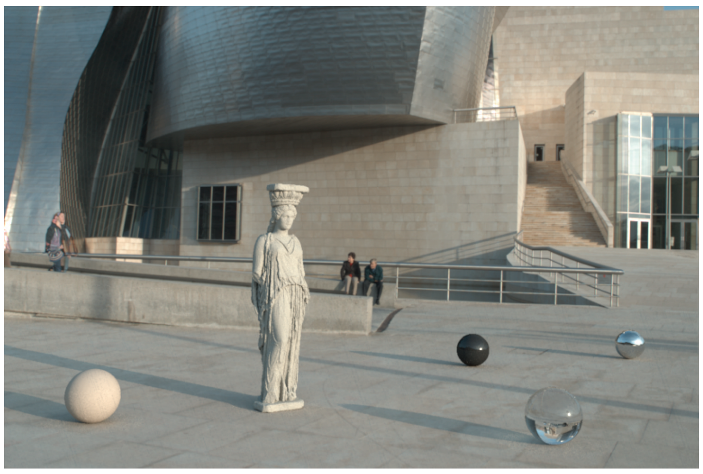


FIGURE 9.38 *Adding synthetic objects that cast shadows. (a) A background plate taken near the location of the light probe image in Figure 9.14. It shows the radiance L of pixels in the local scene. (b) A light probe image taken within the scene. (c) A diffuse surface is added to the scene at the position of the ground and is rendered as lit by the surrounding IBL environment. The resulting pixel values on the surface produce an estimate of the irradiance E at each pixel. (d) Dividing a by c yields the lighting-independent texture map for the local scene. (e) A CG statue and four CG spheres are added on top of the local scene and illuminated by the IBL environment. The CG objects cast shadows and interreflect light with the real scene.*

For example, a red ball on the floor will make the floor around it somewhat redder. Effects such as this are often noticed only subconsciously but can contribute significantly to the realism of rendered images.



(e)

FIGURE 9.38 *(continued)*

The examples we have seen so far from *Rendering with Natural Light* and the Parthenon did not need to simulate shadows back into the scene. The Parthenon model was a complete environment with its own surrounding ground. In RNL, the pedestal received shadows from the objects but the pedestal itself was part of the computer-generated scene. The example we show in this section involves adding several CG objects into the photograph of a museum plaza (seen in Figure 9.38(a)) such that the objects realistically shadow and reflect light with the ground below them (see Figure 9.38(c)). In this example, the light probe image that corresponds to this background plate is the sunny lighting environment shown in Figure 9.14.

Usually, the noticeable effect a new object has on a scene is limited to a local area near the object. We call this area the *local scene*, which in the case of the scene in Figure 9.38(a) is the ground in front of the museum. Our strategy for casting shadows on the local scene is to convert the local scene into a CG representation of its geometry and reflectance that can participate in the lighting simulation along with the CG objects. We can then use standard IBL machinery to light the objects and the local scene together by the IBL environment. The local scene model does not need to have the precise geometry and reflectance of what it represents in the real world, but it does need to satisfy two properties. First, it should have approximately the same geometry that the real scene it represents, in that the shadows reveal some of the scene structure. Second, it should have reflectance properties (e.g., texture maps) that cause it to look just like the real local scene when illuminated on its own by the IBL environment.

Modeling the geometry of the local scene can be done by surveying the scene or using photogrammetric modeling techniques (e.g., Debevec et al. [165]) or software (e.g., ImageModeler by Realviz, *www.realviz.com*). Because the geometry needed is often very simple, it can also be modeled by eye in a 3D modeling package. Similar techniques should also be used to recover the camera position, rotation, and focal length used to take the background photograph.

Obtaining the appropriate texture map for the local scene is extremely simple. We first assign the local scene a diffuse white color. We render an image of the white local scene as lit by the IBL lighting environment, as in Figure 9.38(c). We then divide the image of the real local scene (Figure 9.38(a)) in the background plate by the rendering to produce the reflectance image for the local scene (Figure 9.38(d)). Finally, we texture map the local scene with this reflectance image using camera projection (supported in most rendering systems) to project the image onto the local scene geometry. If we create a new IBL rendering of the local scene using this texture map, it will look just as it did in the original background plate. The difference is that it now participates in the lighting calculation, and thus can receive shadows and interreflect light with synthetic objects added to the scene.

Figure 9.38(e) shows the result of adding a 3D model of a sculpture and four spheres on top of the local scene geometry before computing the IBL rendering. The objects cast shadows on the ground as they reflect and refract the light of the scene. One can also notice bounced beige light near the ground of the beige sphere. The color and direction of the shadows are consistent with the real shadow cast on the ground by a post to the right of the camera's view.

The reason the reflectance-solving process works derives from the fact that the pixel color of a diffuse surface is obtained by multiplying its surface color by the color and intensity of the light arriving at the surface. This calculation was described in detail in case 3 of Section 9.5. More technically stated, the surface's radiance L is its reflectance ρ times the irradiance E at the surface. Inverting this equation, we have $\rho = L/E$. The background plate image shows the surface radiance L of each pixel of the local scene. The IBL rendering of the diffuse white version of the local scene yields an estimate of the irradiance E at every point on the local scene. Thus, dividing L by the estimate of E yields an image ρ of the surface reflectance of the local scene. By construction, assuming the local scene is convex these are the reflectance properties that make the local scene look like the background plate when lit by the IBL environment.

This process can be modified slightly for improved results when the local scene is nonconvex or non-Lambertian. For a nonconvex local scene such as a chair or a staircase, light will reflect between the local scene's surfaces. That means that our estimate of the irradiance E arriving at the local scene's surfaces should account for light from the IBL environment as well as interreflected light from the rest of the local scene. The process described earlier does this, but the indirect light is computed as if the local scene were completely white, which will usually overestimate the amount of indirect light received from the other surfaces. As a result, the local scene's reflectance will be underestimated, and the local scene texture's map will be too dark in concave areas.

The way to avoid this problem is to assume more accurate reflectance properties for the local scene before computing the irradiance image. Typical surfaces in the world reflect approximately 25% of the incident illumination, and thus assuming an initial surface reflectance of $\rho_0 = 0.25$ is a more reasonable initial guess. After rendering the local scene, we should compute the irradiance estimate E_0 as $E_0 = L_0/\rho_0$, where L_0 is the rendering generated by lighting the local scene by the IBL environment. This is another simple application of the $L = \rho E$ formula. Then the texture map values for the local scene can be computed as before, as $\rho_1 = L/E_0$.

If the local scene exhibits particularly significant self-occlusion or spatially varying coloration, the reflectance estimates can be further refined by using the computed reflectance properties ρ_1 as a new initial estimate for the same procedure. We illuminate the new local scene by the IBL environment to obtain L_1, estimate a new map of the irradiance as $E_1 = L_1/\rho_1$, and finally form a new estimate for the local scene's per-pixel reflectance ρ_2. Typically, the process converges quickly to the solution after one or two iterations. In fact, this basic process was used to derive the surface reflectance properties of the Parthenon model seen in Figure 9.28, in which the "local scene" was the entire monument [163].

In some cases, we can simplify the surface reflectance estimation process. Often, the local scene is a flat ground plane and the IBL environment surface is distant or infinite. In this case, the irradiance E is the same across the entire surface, and can be computed as the upward-pointing direction of a diffuse convolution of the light probe image. This eliminates the need to render a complete irradiance image such as shown in Figure 9.38(c), and the local scene reflectance is simply its appearance in the background plate divided by the RGB value E.

9.7.1 DIFFERENTIAL RENDERING

The need to use camera projection to map the local scene texture onto its geometry can be avoided using the differential rendering technique described in [161]. In differential rendering, the local scene is assigned (often uniform) diffuse and specular surface reflectance properties similar to the reflectance of the local scene. These properties are chosen by hand or computed using the reflectance estimation process described previously. Then, two renderings are created: one of the local scene and the objects together (L_{obj}), and one of the local scene without the objects (L_{noobj}). For L_{obj}, an alpha channel image α is created that is 1 for the object pixels and 0 for the non-object pixels, preferably with antialiasing and transparency encoded as gray levels. If L_{noobj} and L_{obj} are the same, there is no shadowing. Where L_{obj} is darker, there are shadows, and where L_{obj} is brighter there are reflections or indirectly bounced light. To apply these photometric effects to the background plate L, we offset its pixel values by the difference between L_{obj} and L_{noobj}. Specifically, the final rendering is computed as

$$L_{\text{final}} = \alpha L_{\text{obj}} + (1 - \alpha)(L + L_{\text{obj}} - L_{\text{noobj}}).$$

In this formula, the α mask allows L_{final} to copy the appearance of the objects directly from L_{obj}, and the local scene is rendered using differential rendering.

As an alternative method, we can apply the *ratio* of L_{obj} and L_{noobj} to the background plate, changing the last term of the formula to $L \times L_{\text{obj}}/L_{\text{noobj}}$. If the reflectance properties of the local scene and the IBL environment are modeled accurately, the background plate L and the local scene lit by the IBL environment L_{noobj} would be the same. In this case, the local scene's appearance in L_{noobj} is copied to L_{final} regardless of whether the difference or the ratio formula is used. When there are inaccuracies in either the lighting or the reflectance, either formula may yield a convincing approximation to the correct result. The difference formula may provide better results for specular reflections and the ratio formula may provide better results for shadows. In either case, only differences between L_{obj} and L_{noobj} will modify the background plate, and where the objects do not affect the local scene it will look precisely as it did in the background plate.

The benefit of this technique is that the local scene does not need to be projectively texture mapped with its appearance in the background plate image. The drawback is that mirror- and glass-like CG objects will not reflect images of the original local scene. Instead, they will reflect images of the modeled local scene.

9.7.2 RENDERING INTO A NONDIFFUSE LOCAL SCENE

If the local scene is somewhat shiny, we would like the new objects to also appear in reflections in the scene. This was the case for *Fiat Lux* (Figure 9.30), where the marble floor of St. Peter's Basilica is notably specular. The problem is compounded by the fact that a shiny local scene may already have visible specularities from bright parts of the lighting environment, and these reflections should disappear when virtual objects are placed between the light sources and the observed locations of their specular reflections. Thus, the synthetic local scene needs to model the specular as well as the diffuse reflection characteristics of the real local scene. Unfortunately, estimating spatially varying diffuse and specular reflection components of a surface, even under known illumination, is usually prohibitively challenging for current reflectometry algorithms.

The easiest procedure to follow is to first manually remove visible specular reflections in the local scene using an image-editing program. In *Fiat Lux*, the notable specular reflections in the floor were from the windows, and in some cases from the lights in the vaulting. Using the edited background plate, we then solve for the local scene reflectance assuming that it is diffuse (as described previously). For the final rendering, we add a specular component to the local scene reflectance whose intensity and roughness are selected by hand to match the appearance of the specularities seen in the local scene on the background plate. The IBL rendering will then show the new CG objects reflecting in the local scene according to the specified specular behavior, and light sources in the IBL environment will also reflect in the local scene when their light is not blocked by the CG objects. This process also provides opportunities for art direction. In *Fiat Lux*, for example, the floor of St. Peter's was chosen to have a more polished specular reflection than it really had, to increase the visual interest of its appearance in the animation.

Sometimes a local scene's specular reflectance dominates its diffuse reflectance, as would be seen for a steel or black marble floor. In these cases, it can be difficult to remove the specular reflection through image editing. In such a case, the best solution may be to model the reflectance of the local scene by eye, choosing specular intensity and roughness parameters that cause reflections of the IBL environment to match the local scene's original appearance reasonably well. If the scene is available for photography under controlled lighting, the local scene can be shaded from specular reflections and illuminated from the side to observe its diffuse component. If the reflectance of the local scene is especially complex and spatially varying, such as an ancient stone and metal inlaid mosaic, one could use a technique such as that described in McAllister [183] or Gardner et al. [169] to derive its surface reflectance parameters by analyzing a set of images taken from many incident illumination directions.

9.8 USEFUL IBL APPROXIMATIONS

Many of today's rendering programs include specific support for image-based lighting (often referred to as *HDRI*), making IBL a straightforward process to use for many computer graphics applications. However, not every production pipeline is designed to support global illumination, and real-time applications require faster rendering times than ray-traced illumination solutions typically allow. Fortunately, there are several approximate IBL techniques that allow particularly fast rendering times and that can be implemented within more traditional rendering pipelines. The sections that follow describe two of them: *environment mapping* and *ambient occlusion*. The discussion includes the advantages and disadvantages of these two approaches.

9.8.1 ENVIRONMENT MAPPING

Environment mapping [154,186,207,171] is a forerunner of image-based lighting in which an omnidirectional LDR image of an environment is directly texture mapped onto an object surface to produce the appearance of it reflecting the environment. The omnidirectional *environment map* or *reflection map* image can also be *pre-convolved* by a blurring filter to simulate the reflections from rough specular or diffuse surfaces. The environment map is mapped onto the object according to each point's surface normal, which makes the rendering process extremely fast once the appropriate reflection map images have been computed. The disadvantage is that the technique does not take into account how light is shadowed and interreflects between object surfaces, which can be an impediment to realism. For objects that are relatively shiny and convex, the errors introduced by the approximation can be insignificant. For objects with more complex geometry and reflectance properties, however, the results can be less realistic.

Environment mapping is most successful and most often used for simulating the specular reflections of an object. In this case, for each surface point the ray from the camera R is reflected about the surface normal N to determine the reflected vector R', computed as follows.

$$R' = R - 2(R \cdot N)N$$

Then, the point on the object is drawn with the pixel color of the environment image corresponding to the direction of R'. Figure 9.39(a) shows this environment mapping process applied to the scene shown in Figure 9.1.

The environment-mapped rendering gives the appearance that the objects are reflecting the environment. However, the appearance is somewhat strange because we do not see reflections of the sphere in the table (the speheres appear to float above it). We can compare this rendering to the corresponding IBL rendering of mirror-like objects in Figure 9.39(b), which exhibits appropriate interreflections. If the scene were a single convex object, however, the two renderings would be the same.

It is interesting to note that this form of environment mapping does not require that the environment map be higher in its dynamic range than the final display. Because every pixel in the rendering comes directly from the environment map, clipping the pixel values of the environment map image would be unnoticeable on

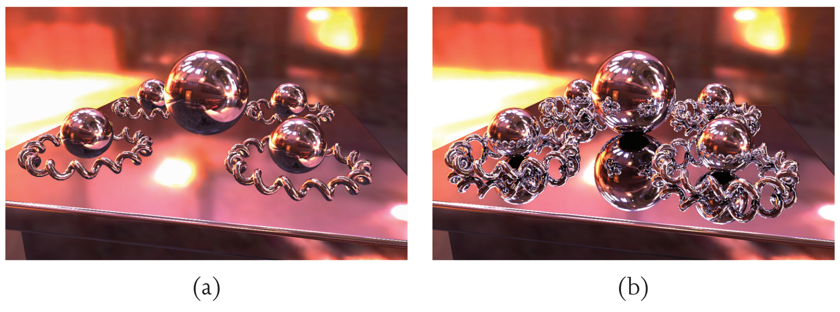


FIGURE 9.39 *(a) A shiny version of the test scene rendered using environment mapping. (b) A shiny version of the scene rendered with ray-tracing-based IBL, producing interreflections.*

a similarly clipped display, unless the rendering were to exhibit significant motion blur or image defocus.

Environment mapping can also be used to simulate the reflection of an environment by surfaces with non-mirror reflectance properties, by *pre-convolving* the image of the environment by various convolution filters [186,171,155,174]. This takes advantage of an effect noted by Ramamoorthi and Hanrahan [195] that a detailed environment reflected in a rough specular surface looks similar to a blurred environment reflected in a mirror-like surface. Often, a specular Phong cosine lobe [194] is used as the convolution filter.

To simulate Lambertian diffuse reflection with environment mapping, a hemispherical cosine lobe is used as the convolution filter, yielding an *irradiance environment map*. For diffuse reflection, one indexes into the irradiance environment map using the object point's surface normal direction N rather than the reflected vector R'. Convolving the image can be computationally expensive, but because irradiance images lack sharp detail a close approximation can be made by convolving a low-resolution version of as few as 32×16 pixels in latitude-longitude format. Cabral et al. [155] suggested that such a convolution could be performed efficiently on a spherical harmonic (SH) decomposition of the incident illumination, and Ramamoorthi and Hanrahan [195] noted that computing the SH reconstruction of an

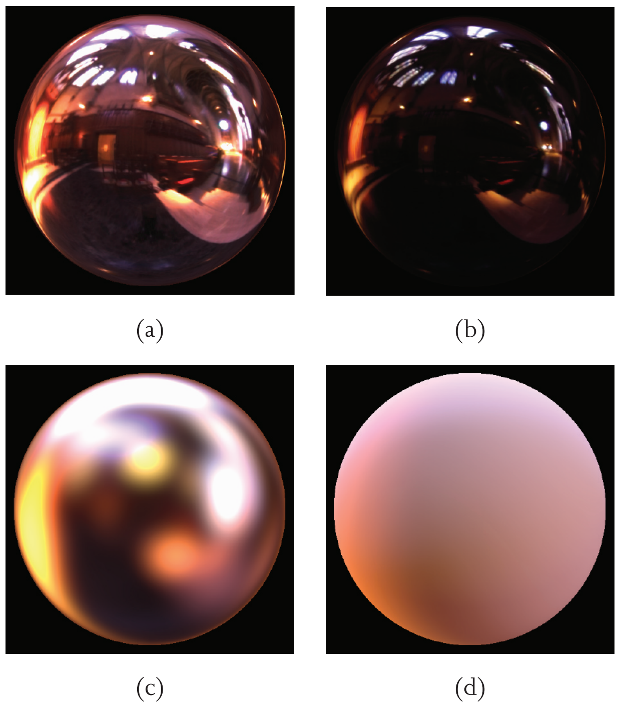


FIGURE 9.40 *(a) HDR Grace Cathedral light probe image, in the mirrored sphere format. (b) The light probe in* **a** *shown with lower exposure, revealing details in the bright regions. (c) A specular convolution of* **a**. *(d) A diffuse convolution of* **a**, *showing how this lighting environment would illuminate a diffuse sphere. (e) An LDR environment map version of* **a** *with clipped pixel values. (f) The image in* **e** *with lower exposure, showing that the highlights have been clipped. (g) A specular convolution of* **e**, *showing inaccurately reduced highlight size and intensity relative to* **c**. *(h) A diffuse convolution of* **e**, *yielding an inaccurately dark and desaturated rendering compared to* **d**.

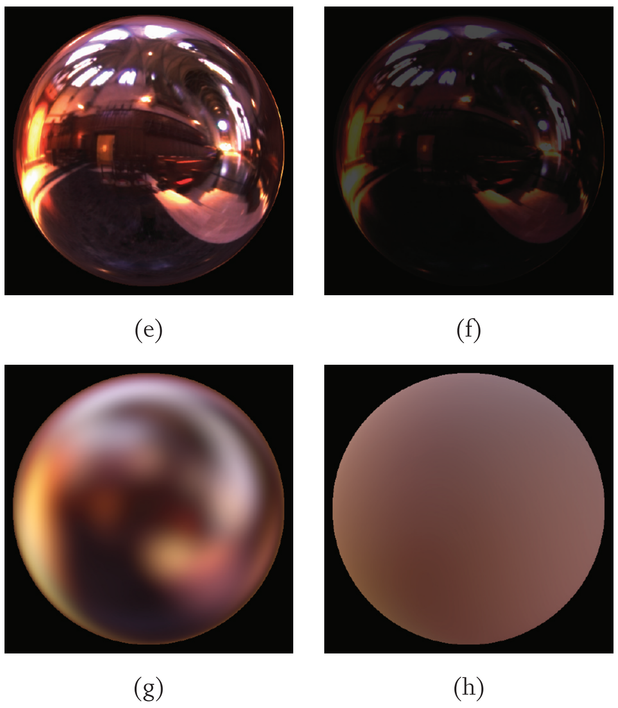


FIGURE 9.40 *(continued)*

environment using the first nine terms (orders 0, 1, and 2) of the SH decomposition approximates the diffuse convolution of any lighting environment to within 99% accuracy.[7]

7 This technique for computing an irradiance environment map can yield regions with negative pixel values when applied to an environment with concentrated light sources due to the Gibbs ringing phenomenon.

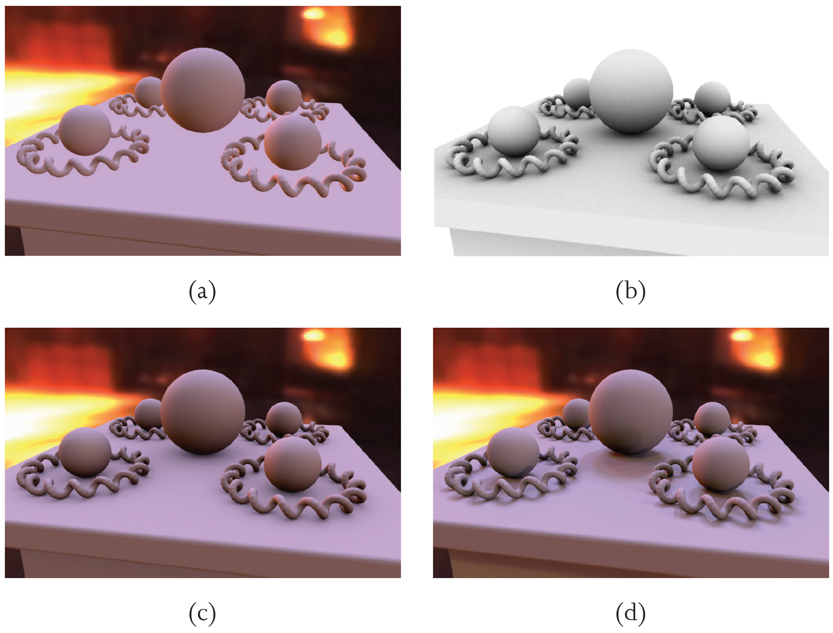


FIGURE 9.41 *(a) A scene environment mapped with the diffuse convolution of the Grace Cathedral environment. (b) An ambient occlusion map obtained using IBL to light the scene with a homogeneous white lighting environment. (c) The ambient occlusion map multiplied by a. (d) The scene illuminated using standard IBL from the light probe image. For this scene, the principal difference from c is in the shadow regions, which have more directional detail in d.*

Figure 9.41 (a) shows a scene rendered using diffuse environment mapping. Because environment mapping does not simulate self-shadowing, the image is not as realistic a rendering of the scene as the IBL solution shown in Figure 9.41 (d). However, if the scene were a single convex object the two renderings would again be the same. In general, environment mapping produces more convincing results for specular reflection than for diffuse reflection. As a result, for objects with both specular and diffuse reflectance components it is common for environment mapping to be used for the specular reflection and traditional lighting for the diffuse component. The technique of *reflection occlusion* [179] can further increase the realism of specular environment mapping by tracing reflected rays from each surface point to determine if the environment's reflection should be omitted due to self-occlusion.

When the incident illumination is blurred by a convolution filter, it becomes necessary that the environment map cover the full dynamic range of the incident illumination to obtain accurate results. Figure 9.40 shows a comparison of using an LDR environment map versus an HDR light probe image for rendering a diffuse sphere using convolution and environment mapping.

9.8.2 AMBIENT OCCLUSION

Ambient occlusion [179] can be used to approximate image-based lighting using a single-bounce irradiance calculation under the assumption that the IBL lighting environment is relatively even. The technique leverages the key IBL step of firing rays out from object surfaces to estimate the amount of light arriving from the visible parts of the environment, but uses diffuse environment mapping to determine the coloration of the light at the surface point. The result is an approximate but efficient IBL process that can perform well with artistic guidance and that avoids noise from the light sampling process.

The first step in ambient occlusion is to use an IBL-like process to render a neutral diffuse version of the scene as illuminated by a homogeneously white illumination environment. In this step, the surfaces of the scene are set to a neutral diffuse reflectance so that the rendered image produces pixel values that are proportional to a surface point's irradiance. This step can be performed using the Monte Carlo ray-tracing process (described in Section 9.5) or by converting the white lighting environment into a constellation of light sources (Section 9.6.2). In the latter case, the rendering can be formed by adding together scan-line renderings of the scene lit from each lighting direction, with the shadows being calculated by a shadow buffer algorithm. Usually, no additional light bounces are simulated when computing this rendering. This estimate of the irradiance from the environment at each point is called the *ambient occlusion map*, seen in Figure 9.41(b).

The next step is to multiply the ambient occlusion map by an image of the scene environment mapped with the diffuse convolution of the lighting environment, as in Figure 9.41(a). The product, seen in Figure 9.41(c), applies the self-shadowing characteristics of the ambient occlusion map to the diffuse lighting characteristics of the environment-mapped scene, yielding a rendering that is considerably more realistic. If the scene has different surface colors, this rendering can be multiplied by an image of the diffuse color of each point in the scene, which approximates having differently colored surfaces during the original rendering. If needed, specular reflections can be added using either ray tracing or specular environment mapping.

Ambient occlusion does not precisely reproduce how a scene would appear as illuminated by the light probe using standard IBL. We can see, for example, differences in comparing Figures 9.41(c) and 9.41(d). Most notably, the shadows in the ambient occlusion rendering are much softer than they appear in the standard IBL rendering. The reason is that the ambient occlusion rendering is computed as if from a completely diffuse lighting environment, whereas a standard IBL rendering computes which specific parts of the lighting environment become occluded for each part of the shadowed area. The ambient occlusion result can be improved to some extent using *bent normals* [179], where the diffuse convolution of the light probe image is mapped onto the object surfaces according to the average direction of unoccluded light, rather than the true surface normal. However, because the surface colors are still sampled from a diffuse convolution of the light probe image, the ambient occlusion rendering will lack the shading detail obtainable from sampling the light probe image directly.

Ambient occlusion most accurately approximates the correct lighting solution when the lighting environment is relatively diffuse. In this case, the homogeneous environment used to compute the occlusion is a close approximation to the environment desired to light the scene. Ambient occlusion is not designed to simulate light from environments that include concentrated light sources, as the directional detail of the environment is lost in the diffuse convolution process. For IBL environments that do have concentrated light sources, an effective way of handling them is to simulate them as direct light sources (as described in Section 9.6.1), delete them from the IBL environment, and use a diffuse convolution of the modified IBL environment to multiply the ambient occlusion map.

Although computing ambient occlusion maps requires sending out a multitude of rays to the lighting environment, the number of rays that need to be sent is minimized because the environment has minimal variance, which alleviates the sampling problem. Also, the ambient occlusion map is solely a function of the object geometry and is independent of the lighting environment. Because of this, the technique can be used to render an object with different lighting environments while performing the ambient occlusion calculation map only once. This makes real-time implementations very fast, especially for rotating lighting environments for which performing additional diffuse convolutions is also unnecessary. In addition, the technique allows for relighting effects to be performed inside a standard compositing system. For example, the convolved light probe image can be manually edited and a relit version of the scene can be created quickly using the preexisting normals and ambient occlusion map without rerendering.

9.9 IMAGE-BASED LIGHTING FOR REAL OBJECTS AND PEOPLE

The IBL techniques described so far are useful for lighting synthetic objects and scenes. It is easy to imagine uses for a process that could illuminate *real* scenes, objects, and people with IBL environments. To do this, one could attempt to build a virtual model of the desired subject's geometry and reflectance and then illuminate the model using the IBL techniques already presented. However, creating photoreal models of the geometry and reflectance of objects (and particularly people) is a difficult process, and a more direct route would be desirable. In fact, there is a straightforward process for lighting real subjects with IBL that requires only a set of images of the subject under a variety of directional lighting conditions.

9.9.1 A TECHNIQUE FOR LIGHTING REAL SUBJECTS

The technique is based on the fact that light is *additive*, which can be described simply as follows. Suppose we have two images of a subject, one lit from the left and one lit from the right. We can create an image of the subject lit with both lights at once simply by adding the two images together, as demonstrated by [172]. If the image pixel values are proportional to the light in the scene, this process yields exactly the right answer, with all of the correct shading, highlights, and shadows the scene would exhibit under both light sources. Furthermore, the color channels of the two images can be independently scaled before they are added, allowing one to virtually light the subject with a bright orange light to the right and a dim blue light to the left, for example.

As we have seen in Section 9.6.2, an IBL lighting environment can be simulated as a constellation of light sources surrounding the subject. If one could quickly light a person from a dense sampling of directions distributed across the entire sphere of incident illumination, it should be possible to recombine tinted and scaled versions of these images to show how the person would look in any lighting environment. The Light Stage device described by [162] (Figure 9.42) is designed to acquire precisely such a data set. The device's 250-watt halogen spotlight is mounted on a two-axis rotation mechanism such that the light can spiral from the top of the sphere to the bottom in approximately one minute. During this time, a set of digital video cameras can record the subject's appearance as illuminated by hundreds of lighting directions distributed throughout the sphere. A subsampled light stage data set of a person's face is seen in Figure 9.43(a).

Figure 9.43(c) shows the Grace Cathedral lighting environment remapped to be the same resolution and in the same longitude-latitude space as the light stage data set. For each image of the face in the data set, the remapped environment indicates the color and intensity of the light from the environment in the corresponding direction. Thus, we can multiply the red, green, and blue color channels of each light stage image by the amount of red, green, and blue light in the corresponding direction in the lighting environment to obtain a modulated image data set, as in Figure 9.43(d). Adding all of these images together then produces an image of the subject as illuminated by the complete lighting environment, as seen in Figure 9.44(a). Results obtained for three more lighting environments are shown in Figures 9.44(b) through 9.44(d).

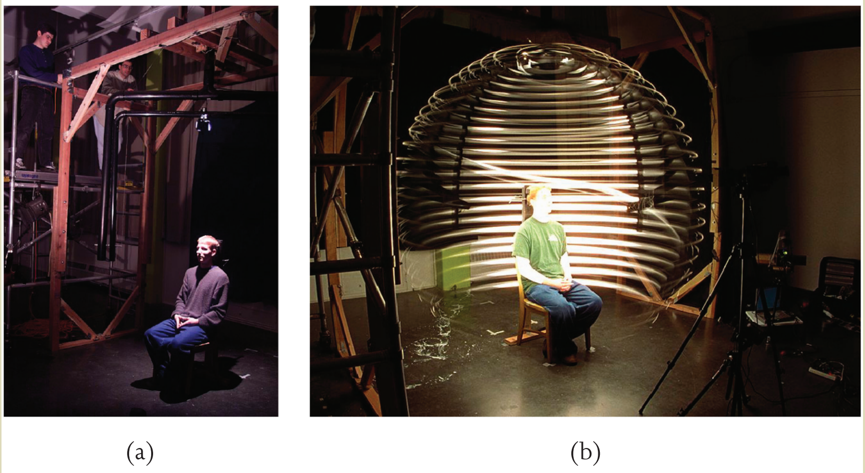


FIGURE 9.42 *(a) The Light Stage 1 device for lighting a person's face from the full sphere of incident illumination directions. (b) A one-minute exposure taken during a data set acquisition. Images recorded from the right video camera are shown in Figure 9.43.*

9.9.2 RELIGHTING FROM COMPRESSED IMAGE DATA SETS

Computing the weighted sum of the light stage images is a simple computation, but it requires accessing a large amount of data to create each rendering. This process can be accelerated by performing the computation on compressed versions of the original images. In particular, if the images are compressed using an orthonormal transform such as the discrete cosine transform (DCT), the linear combination of the images can be computed directly on the basis coefficients of the compressed images [197]. The downloadable Facial Reflectance Field Demo [203] (*www.debevec.org/FaceDemo/*) uses DCT-compressed versions of light stage data sets to allow a user to interactively relight a face using either light probe images or user-controlled light sources in real time.

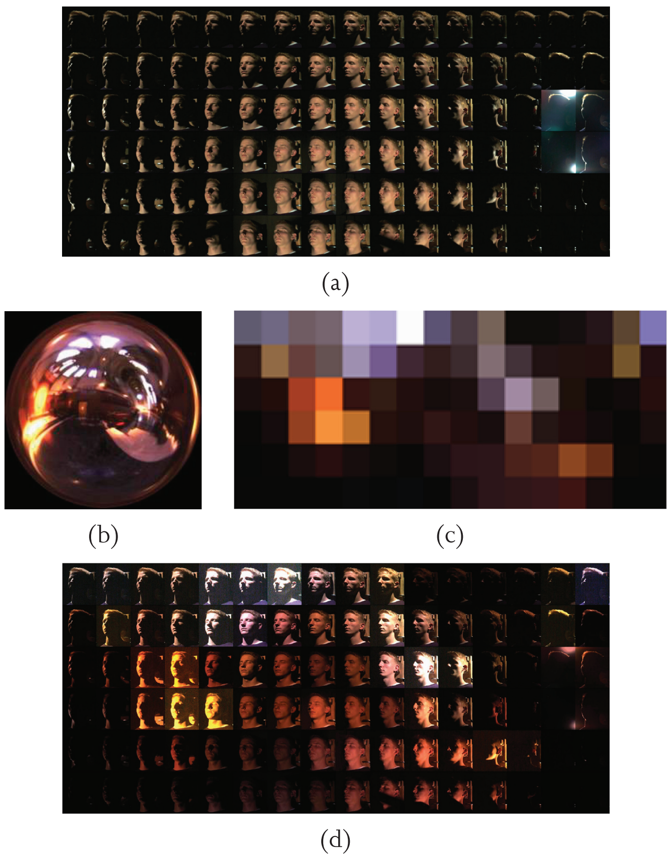


FIGURE 9.43 *(a) Light stage data of a face illuminated from the full sphere of lighting directions. The image shows 96 images sampled from the 2,000-image data set. (b) The Grace Cathedral light probe image. (c) The Grace probe resampled into the same longitude-latitude space as the light stage data set. (d) Face images scaled according to the color and intensity of the corresponding directions of illumination in the Grace light probe. Figure 9.44(a) shows the face illuminated by the Grace probe created by summing these scaled images.*

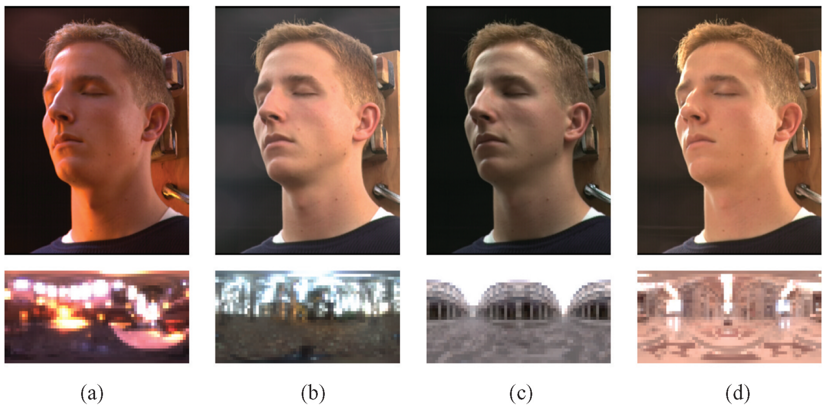


FIGURE 9.44 *Renderings of the light stage data set from Figure 9.43 as illuminated by four image-based lighting environments: (a) Grace cathedral, (b) Eucalyptus grove, (c) Uffizi gallery, and (d) St. Peter's Basilica.*

A light stage data set can be parameterized by the four dimensions of image coordinates (u, v) and lighting directions (θ, ϕ). Choosing a particular pixel (u, v) on the subject, we can create a small image (called the pixel's *reflectance function*) from the color the pixel reflects toward the camera for all incident lighting directions (θ, ϕ) (Figure 9.45). In the Facial Reflectance Field Demo, the 4D light stage data sets are actually DCT compressed in the lighting dimensions rather than the spatial dimensions, exploiting coherence in the reflectance functions rather than in the images themselves. When the DCT coefficients of the reflectance functions are quantized (as in JPEG compression), up to 90% of the data maps to zero and can be skipped in the relighting calculations, enabling real-time rendering. The process of relighting a single pixel of a light stage data set based on its reflectance function is shown in Figure 9.45.

This image-based relighting process can also be applied in the domain of computer-generated objects. One simply needs to render the object under an ar-

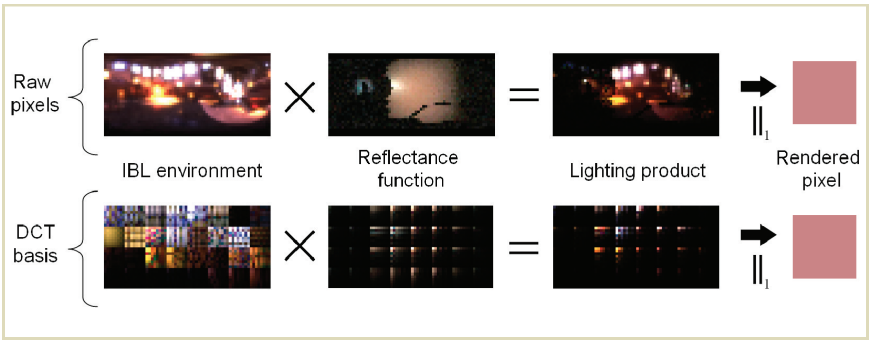


FIGURE 9.45 *Relighting a reflectance function can be performed on the original pixel values (top row) or on DCT coefficients of the illumination data.*

ray of different lighting conditions to produce a virtual light stage data set of the object. This can be useful in that the basis images can be rendered using high-quality offline lighting simulations but then recombined in real time through the relighting process, maintaining the quality of the offline renderings. In the context of CG objects, the content of a reflectance function for a surface point is its *precomputed radiance transfer*. Sloan et al. [196], Ramamoorthi and Hanrahan [195], and Ng et al. [189] have noted that the basis lighting conditions need not be rendered with point source illumination. Specifically, Sloan et al. [196] and Ramamoorthi and Hanrahan [195] use the Spherical Harmonic (SH) basis, whereas Ng et al. [189] use a wavelet basis. These techniques demonstrate varying the viewpoint of the object by mapping its radiance transfer characteristics onto a 3D geometric model of the object. Whereas these earlier techniques have been optimized for diffuse surface reflectance in low-frequency lighting environments, Liu et al. [182] use both a wavelet representation and clustered principal components analysis of PRT functions to render view-dependent reflections from glossy objects in high-frequency lighting environments, producing sharp shadows from light sources interactively. Sample renderings made using these techniques are shown in Figure 9.46. Using the GPU to perform image-based relighting on CG objects with techniques such as

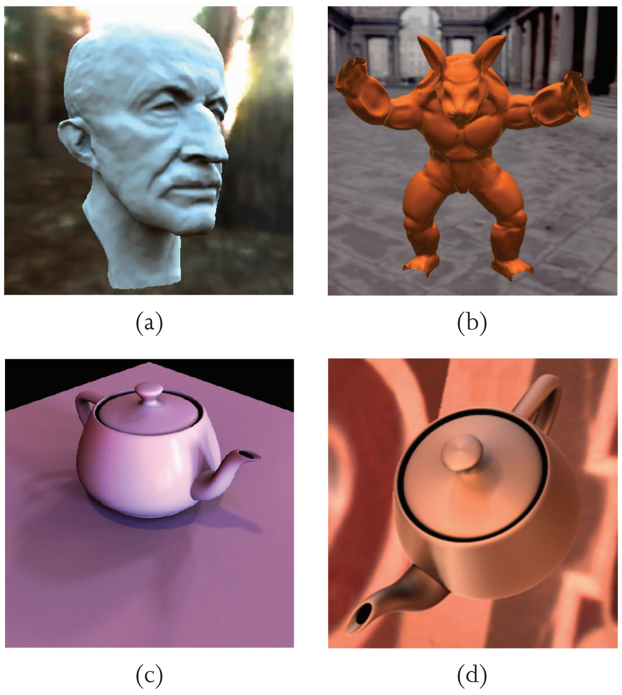


FIGURE 9.46 Interactive IBL renderings of 3D objects using basis decompositions of the lighting environment and surface reflectance functions. (a) Max Planck model rendered in the RNL environment using precomputed radiance transfer [196] based on spherical harmonics. (b) Armadillo rendered in the Uffizi Gallery using a spherical harmonic reflection map [195]. (c) Teapot rendered into Grace Cathedral using a 2D Haar transform [189] of the lighting and reflectance, achieving detailed shadows. (d) Teapot rendered into St. Peter's Basilica using precomputed radiance transfer represented by Haar wavelets and compressed with clustered principal component analysis [182] to produce sharp shadow detail, as seen on the teapot lid.

these promises to become the standard method for using IBL in video games and other interactive rendering applications.

9.10 CONCLUSIONS

In this chapter we have seen how HDR images can be used as sources of illumination for both computer-generated and real objects and scenes through image-based lighting. Acquiring real-world lighting for IBL involves taking omnidirectional HDR images through one of several techniques, yielding a data set of the color and intensity of light arriving from every direction in the environment. This image of the incident illumination is mapped onto a surface surrounding the object, and a lighting simulation algorithm is used to compute how the object would appear as if lit by the captured illumination. With the appropriate optimizations, such images can be computed efficiently using either global illumination or traditional rendering techniques, and recent techniques have allowed IBL to happen in real time. Real objects can be illuminated by new environments by capturing how they appear under many individual lighting conditions and then recombining them according to the light in an IBL environment.

The key benefit of IBL is that it provides a missing link between light in the real world and light in the virtual world. With IBL, a ray of light can be captured by an HDR camera, reflected from a virtual surface in a rendering algorithm, and be turned back into real light by a display. The IBL process has a natural application wherever it is necessary to merge CG imagery into real scenes, in that the CG can be lit and rendered as if it were actually there. Conversely, the light stage technique allows IBL to illuminate real-world objects with the light of either virtual or real environments. In its applications so far, IBL has rendered virtual creatures into real movie locations, virtual cars onto real roads, virtual buildings under real skies, and real actors into virtually created sets. For movies, video games, architecture, and design, IBL can connect what is real with what can only be imagined.

ACKNOWLEDGEMENTS

In regard to Chapter 9, we wish to thank Andrew Jones, Chris Tchou, Andrew Gardner, Tim Hawkins, Andreas Wenger, Per Einarsson, Maya Martinez, David

Wertheimer, Bill Swartout, and Lora Chen for their help during the preparation of this chapter. The Parthenon model was created with the additional collaboration of Brian Emerson, Marc Brownlow, Philippe Martinez, Jessi Stumpfel, Marcos Fajardo, Therese Lundgren, Nathan Yun, CNR-Pisa, Alias, and Geometry Systems. Portions of the work described in this article have been sponsored by a National Science Foundation Graduate Research Fellowship, a MURI Initiative on 3D direct visualization from ONR and BMDO (grant FDN00014-96-1-1200), Interval Research Corporation, the University of Southern California, and the U.S. Army Research, Development, and Engineering Command (RDECOM).