Combining Dynamic Physical and Virtual Illumination in Augmented Reality

Stephen DiVerdi Tobias Höllerer

Department of Computer Science
University of California, Santa Barbara, CA 93106

Abstract

With rapidly advancing graphics hardware available, real-time graphics techniques have become significantly more capable of handling photorealistic effects. The application of these techniques to augmented reality is creating new possibilities for the seamless integration of virtual and physical objects. We demonstrate two methods for simulating light transport. The first renders virtual objects of arbitrary Phong material with light from the physical environment. The second adds virtual light onto real-time camera imagery of the physical world. Both techniques yield fully dynamic results at interactive framerates with a minimum of offline preparation.

Key words: Augmented reality, consistent illumination, image-based shading

1 Introduction

One of the goals of augmented reality research is to provide a user experience in which virtual objects are perceived as coexisting with the physical world. A hard problem to address towards this goal is maintaining consistent geometric registration between the virtual and physical geometry - in order to look real, a virtual object must stay aligned with the physical geometry, unless the user is specifically manipulating it. Now that the field is a bit more developed and research into geometric registration is well underway, the second hard problem of seamlessly integrating virtual and physical worlds is real-time photorealistic rendering.

The majority of today's augmented reality applications are wrought with wireframe geometry and Gouraud shaded polygons. Lighting is handled completely by basic models in standard 3D graphics APIs. The differences between virtual and physical objects are very pronounced because the illumination, materials, and visual quality are radically different.

In recent years, research has begun to address this problem. For static environments, with considerable amounts of preparation using both human time and compute cycles, photorealistic results are achievable at low framerates [10, 13]. Precapturing of the lighting environment can yield realistically illuminated virtual geometry [1], and extensive knowledge of environment geometry can even produce virtual shadowing of physical objects [10].

Figure 1: A virtual teapot and spotlight. The teapot has a gold metal material and is lit by both the environment and a blue spotlight overhead. The spotlight also illuminates the physical table below.

Two major problems remain unaddressed however. One is the dynamic capture of illumination data for real-time rendering in a dynamic lighting environment. Most current techniques require that lighting data is acquired beforehand, so rendering is tied to a static set of lights [10, 1]. If the actual environment illumination changes during rendering, the virtual objects will no longer appear to be illuminated by the physical world. To address this problem, we dynamically capture an environment map from a small silvered sphere placed in the scene, and use this dynamic data for rendering in real-time. The acquired environment map can be used as is for mirror reflections, or it can be filtered to provide diffuse and glossy reflections. The result is virtual geometry that will respond appropriately in real-time to changes in the physical lighting.

The second open issue is dynamic virtual lighting of the physical environment. Current state of the art uses detailed knowledge of the environment geometry to cast shadows from virtual occluders onto the physical environment in real-time [10], or, with a significantly lower framerate, a complete global illumination solution can account for all virtual and physical light transfer [13]. Our architecture takes very limited scene geometry information and casts direct lighting from virtual objects, to provide a reasonable approximation of interaction of the virtual light with the surrounding physical scene. A programmable fragment shader enables the real-time calculation of of this lighting information. These simplifications allow us to achieve thirty frames per second on commodity graphics hardware. To create the effect additional lighting would have on the brightness of the camera's image, we post-process the video frame to simulate a shorter exposure, darkening the scene.

The contribution of this paper is a single system that achieves the effects of consistent illumination in real-time, with the flexibility to respond to dynamic changes in the physical environment. We implemented this system as an extension to ARWin [7], an augmented reality research architecture which allows for quick prototyping and testing of new concepts in applications, interfaces and rendering techniques. ARWin uses the ARToolKit [16] for registration and marker-based interaction, to manage the manipulation and display of generic 3D applications in the volume over an office desk.

2 Related Work

While there is not a clear-cut answer to the issue of how to make a virtual object appear to have greater presence in the physical world, a number of studies have been conducted to address this very question. The general consensus is that while visual realism may not be related to productivity in virtual environments, it does increase the virtual world's sense of presence [23, 25]. Certain realistic rendering effects such as shadows have been studied in particular, to show their importance in determining the presence of virtual objects [19, 21]. However, research into perceptually based rendering has shown that there are certain visual effects that humans are insensitive to [8]. Clearly, there is still a need for further study of the effects of realism on user's perception of virtual and augmented reality, but there is a significant amount of evidence suggesting enhanced realism gives the user a greater sense of presence of virtual objects in the real world.

In the past few years, there has been a significant amount of research into advanced lighting and rendering techniques which are applicable to augmented reality. In 1998, Debevec raised interest in the concept of image-based lighting [6], using an image of a light probe to capture environment illumination and then calculate shading based on this data in offline renderers. This concept adapted the much older technique of sphere-mapping, originally introduced in 1976 [4]. Since then, the technique has been adapted to include filtering – integrating the incident light with BRDF material properties to produce environment maps for arbitrary material reflectances [5, 12].

Environment maps have been directly applied in a handful of augmented reality systems. At SIGGRAPH '96, State *et al* [20] demonstrated a video of a silvered sphere morphing into a teapot, environment mapped by the captured video of the sphere. In more recent work, Agusanto *et al* [1] acquire detailed high dynamic range environment maps and filter them offline. The filtered maps are blended together at run time to produce various material properties. Kanbara and Yokoya [11] take a different approach, dynamically acquiring environment map data to control a distribution of point lights around the virtual geometry for diffuse lighting. Both groups produce shaded virtual objects in real-time, but are limited either by inability to respond to dynamic changes in lighting, or inability to simulate many different material types.

Global illumination techniques have also been applied to photorealistic rendering of virtual objects in real scenes. Fournier *et al* proposed an early system in 1993 [9], which would create a detailed scene model and use a global illumination algorithm to add virtual objects and lights. Because of the complex lighting calculations, each frame took minutes to render. The technique was improved by Loscos *et al* in 2000 [13] with the addition of an incremental radiosity algorithm that sped up rendering times to seconds per frame. Gibson and Murta [10] kept real-time rendering a high priority and replaced a global illumination solution with a number of fast approximations for simulating lighting and shadows. The shift away from an accurate rendering solution limits their system to a subset of actual light transport calculations - specifically, the paper does not address light transport from virtual objects to the surrounding physical environment.

Finally, a different approach to the problem of illuminating the physical world has been proposed by Raskar *et al* [18] and Bimber *et al* [3]. Both groups use multiple projectors and careful calibration to project real light onto a real scene, to directly create virtual illumination effects on the physical geometry. The technique can create convincing lighting and shadows based on the presence of virtual objects and light sources. The work presented by these projects is most applicable to hybrid projector-based and optical see-through augmented reality systems.

Figure 2: An ARToolKit marker with our light probe attached. The bullseye shows the marker has been recognized.

3 Physical Illumination

Physical illumination of virtual objects is achieved using image based lighting techniques pioneered by Debevec [6]. The general technique is to acquire an image of a light probe of the scene, and then process this light probe and apply it as a texture map to geometry to create various material responses.

3.1 Light Probe

Our light probe consists of a small (two and three quarter inch diameter) silver sphere (a christmas ornament) mounted on a marker (see Figure 2). For a desktop setup, this could be a small paperweight or decoration. Calibration of the light probe consists of measuring the sphere's position with respect to the marker's center. The screen position of the sphere can be determined from the transformation matrix for the marker as reported by the ARToolKit [16], and these pixels can be copied from the video stream into a texture.

The texture then contains a typical environment map, which can be applied using regular OpenGL sphere mapping for mirror reflections of the environment. For more general rendering of different materials, the environment map texture must be filtered.

3.2 Filtering

Techniques have already been established to process environment maps to produce different material properties. One possibility is to integrate the map with an arbitrary BRDF to produce a map for an arbitrary shaded material [12]. This technique increases in computation time with the number of different materials. A less complex option is to integrate the map with a standard Phong illumination model BRDF to create a set of varying glossy maps, which can be blended at runtime [10]. However, this still requires a significant amount of preprocessing.

In order to be able to dynamically update the filtered environment maps, we adopted a technique from Ashikhmin and Ghosh [2] that creates a rough estimate of Phong integrated environment maps by using OpenGL's built-in mipmap generation capabilities. Before the environment map texture is created, automatic mipmap generation is enabled to quickly create smaller, box filtered versions of the environment map. We can then select which level to render with by specifying the minimum mipmap level. This creates a rough approximation of a glossy material (see Figure 3).

With a set of filtered environment maps, ranging from specular to diffuse, we can simulate arbitrary material properties by selecting the appropriate maps and blending them together. The material's shininess factor determines which level of glossy specular map is blended with the diffuse map. Samples from these textures modulated by the material's diffuse and specular responses represent the material's physical reflectance, which is then added to any virtual lighting calculations and material texture maps for the final reflectance value.

3.3 Discussion of Physical Illumination

While this technique is not physically accurate, it does produce convincing results (see Figure 4). Artifacts arise from inaccuracies in the sphere tracking, which result in some jitter in the acquired environment map. Registration could be improved many ways - better calibration, edge finding for image-space extraction, or even combining maps over a number of frames - to reduce this effect.

The technique used for simulating more diffuse material properties causes an unusual effect around the edges of the environment map - the background pixels are blended in, subtly changing the color at silhouette edges. This effect could be removed by masking out the non-

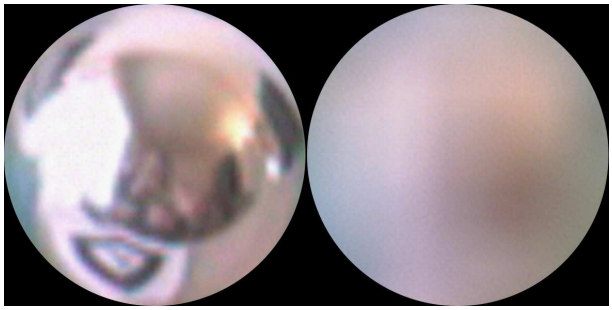

Figure 3: Left: The acquired specular environment map. Right: The filtered environment map for diffuse lighting.

Figure 4: A virtual torus with an unpolished gold material, illuminated by the physical environment.

sphere pixels from the map, but then the black values would be filtered in, darkening the environment map. The best solution would be to use a more accurate filtering technique, such as suggested by Ramamoorthi and Hanrahan [17] for converting the environment map into spherical harmonics and performing the filtering quickly in frequency space. This would also produce more accurate diffuse reflections.

We are also limited by the fundamental assumptions of environment mapping. For example, there is no self-reflection. Environment maps also assume the environment is in the distance, but for our setup generally on or near a desk, objects in the environment map are relatively nearby. This means inaccuracies in reflections become apparent for relatively small virtual object motion. This issue could be addressed by sweeping the silver sphere through the environment or by using multiple silver spheres. Multiple light probe images could then be combined using a combination of warping and blending as in [5].

4 Virtual Illumination

While a conventional AR application does not in general produce light to illuminate the surroundings, it is easy to imagine applications were such a visual would be appropriate. Household objects that produce light include devices with backlights, such as calculators or watches - a virtual digital watch could be made to become bright and produce light when in active mode. A virtual lamp object could be used to provide extra lighting in areas where environment lighting is low, given a common illumination model. However, creating a brighter area for virtual objects without also illuminating the nearby physical environment creates a serious gap in the perception of integrated virtual and physical objects. In order to create a convincing illusion of reality for the virtual objects, it is desirable that the virtual light emitted also illuminates the physical objects nearby.

The added virtual light should also affect the exposure of the camera image. Simple brightness adjustment does not account for saturated regions that appear in low-dynamic range images. Instead, we modify the brightness adjustment based on the saturation of a region, so over-exposed regions darken gradually rather than uniformly.

4.1 Assumptions

The obvious way to add lighting to physical geometry is to create a replica of the physical objects in the virtual world, calculate the lighting for this proxy, and then add the difference into the image of the environment. An exact solution requires a detailed scene geometry model that takes a large amount of time and effort to acquire, plus a complex global illumination renderer to properly simulate light transfer to this geometry. However, the priorities for our system are as little setup as possible and a fully dynamic and real-time user experience, which makes an exact solution prohibitive. Instead, we choose to approximate the ideal result by reducing the scope of the problem in two ways.

First, rather than calculate a full global illumination model of the light transfer, we rely on the regular OpenGL Phong lighting model, simulating only the direct illumination of nearby geometry. With proper use of attenuation and color attributes, a reasonable approximation of a physical omnidirectional or directed point light source can be created. Current graphics hardware can already handle many light sources of this type in real-time with little effort, and avoiding a costly global illumination solution ensures an interactive user experience. Shadows are generated using a basic shadow map [24] algorithm, which is also suitable for real-time rendering.

Second, acquiring detailed scene geometry can be a lengthy and difficult process, and it must be repeated whenever the environment changes. To address this issue, our proposed technique approximates lighting of the environment using a significantly simplified model of the physical objects present. For example, the surface of a desk is modelled as a flat plane, and low-profile objects on the desk such as a mouse or keyboard do not need additional modelling. A monitor can be approximated with a single scaled cube. In fact, for most large objects, a simple scaled box is sufficient – small objects generally do not need modelling. Given the low complexity of this sort of rough scene geometry, it is a much simpler task to create a proxy of the physical environment in short order.

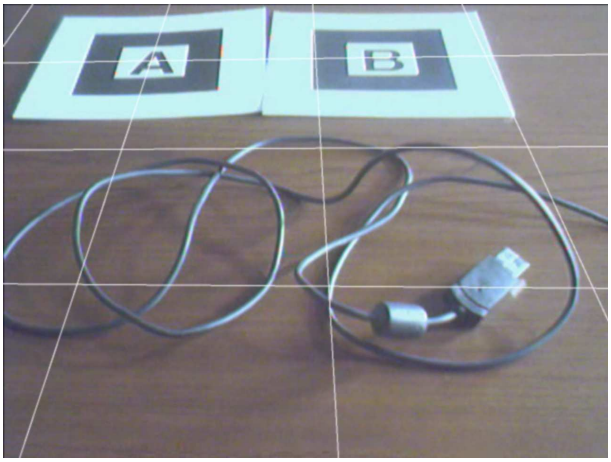

Figure 5: Example physical scene containing a table, modeled by a ground plane. The wire sitting on the table is a complex object that is not included in the model.

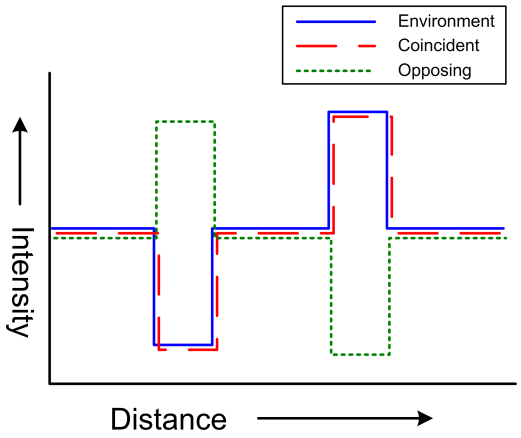

Figure 6: For a given environmental light direction, the intensity profile includes shadows and highlights around bumps. A coincident virtual light source should have the same shadows and highlights, while an opposing virtual light should have the inverse.

4.2 Technique

Simulation of virtual illumination must calculate the contribution from a virtual light source to the reflectance of a bumpy physical surface. Flat polygons are used to approximate rough geometry, so we would like the virtual illumination to properly account for the small fluctuations in height. Simple OpenGL blending creates the appearance of illumination of a flat surface, and so is not sufficient for our purposes. However, the image of the scene captured from the camera provides useful information when projected on top of the simple scene model. Each polygon in the model has a physically lit texture associated with it. If the physical illumination is known then the pattern of highlights and shadows on the texture can be associated with the direction of brightest light. This information represents the bumps along that particular direction. If the scene is ambiently lit, no bump information can be extracted.

Given knowledge of bumps along one direction, we can modify the virtual illumination to take them into account. If the virtual light is placed near the area of brightest physical illumination, then we know that the virtual light will have the same highlights and shadows as are present in the projected video texture. Conversely, if the virtual light placed opposite the brightest physical illumination, the highlights and shadows will be opposite. Since we only know the physical bump data along one axis (the direction of brightest illumination), if the virtual light is placed orthogonal to this axis, we have no extra bump information.

Therefore, we can calculate a more accurate bump-mapped virtual illumination using the projected video texture. We call the flat polygon shaded value the "additive factor", and the intensity of the projected video texture the "multiplicative factor". We then use the dot product of the virtual light vector and the brightest physical lighting vector to determine the contribution of each of these factors. When the dot product is close to one, most of the contribution is from the multiplicative factor, when it is close to zero, the contribution is from the additive factor, and when it is negative one, the inverse of the multiplicative factor is used (see Figure 6).

To render the virtual illumination onto the video texture of the physical environment, we enable the virtual light objects and pass to OpenGL the simplified scene geometry. We perform the blend computations in custom vertex and fragment shaders, written in NVIDIA's Cg shader language [15]. Without programmable shader capabilities, this technique would not be possible in hardware, in which case we would have to resort to alpha blending or a software technique. The shaders are reproduced in pseudocode here.

```
VertexShader {
  VertColor = attenuation * Kd
  TexCoord0 = screen space position
  TexCoord1 = normal vector
  TexCoord2 = light vector
  TexCoord3 = dot(env light vector,
      light vector)
}

FragmentShader {
  video = sample(VideoTex,TexCoord0)
```

```
  alpha = TexCoord3
  additive = dot(TexCoord1,TexCoord2)
  multiplicative = intensity(video)
  response = (1.0-alpha)*additive +
      alpha*multiplicative
  FragColor = video +
      response*VertColor
}
```

Vertex shaders are executed once per rendered vertex in the rendering pipeline. The current OpenGL state is passed in, including material properties and light properties. We manually pass in the environment light vector as well. This data is then interpolated by the rasterizer across the polygon for a particular fragment. To render a fragment, the fragment shader is called with the interpolated values. Our fragment shader samples the video texture to project the video onto the geometry. The additive factor is calculated via diffuse Phong shading, while the multiplicative factor is obtained from the video's intensity. The response of the material is then the interpolation between these factors, and the final color is the video sample plus the attenuated response times the material's diffuse reflectance.

4.3 Exposure Adjustment

Exceptionally bright physical objects, such as light sources or highly reflective materials, appear in low dynamic range images as uniformly saturated pixels. To simulate the effect of a decreased camera exposure time, brightness adjustment treats these pixels as having the same value, when the reality is they cover a wide range of intensities. To better simulate exposure adjustment of a low dynamic range image, we adjust the factor by which intensities are scaled, based on the average intensity of surrounding pixels. The effect is that the center of overexposed regions will remain bright while the edges fade with the rest of the image (see Figure 7). Extreme darkening will eventually decimate the entire saturated region.

To quickly compute average intensities of large regions, an integral image is first computed. This allows the sum of an arbitrary region of pixels to be computed with four samples. The size of the region is determined by the amount of darkening necessary, from 1x1 for 100 percent, linearly up to a maximum region limit at 0 percent.

4.4 Discussion

Results of this algorithm compared to unenhanced OpenGL alpha blending, can be seen in Figure 8 and Figure 9, for a virtual light coinciding with and opposing the physical illumination, respectively.

Our technique for combining virtual illumination with the physical scene suffers from many of the same limitations as bump-mapping. Most notably, from shallow angles the illumination looks as if it were flatly applied on top of the real geometry, instead of being displaced by the small fluctuations in height as it should be. The approximate scene geometry model handles large changes in height to reduce this artifact - therefore, a more accurate rendering can be created by increasing the complexity of the scene geometry.

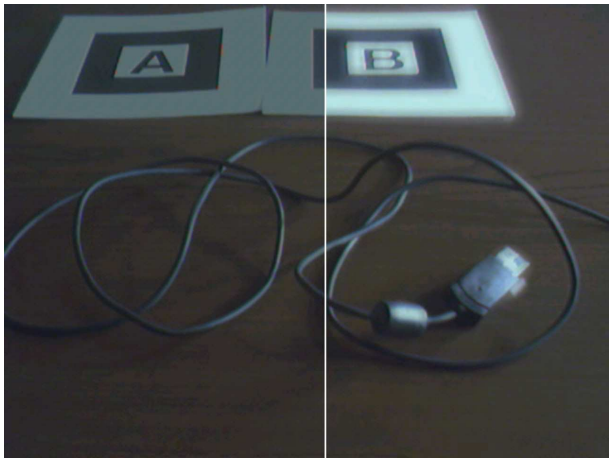

Figure 7: A scene darkened to 50 percent. The left half simply adjusted the image brightness. The right half uses our algorithm.

This technique provides a major advantage in augmented reality however, since the bumped illumination data is acquired from the projected video texture. The algorithm is designed to light unmodeled surface roughness, so illumination of minor changes in scene geometry are accommodated without any additional work. If a simple scene that consists of a flat surface like a table has a small object such as a pencil placed on it, the lighting will automatically highlight and shadow the pencil's geometry without requiring an update to the scene model. This is very useful for dynamic environments like a desk, where small objects are regularly moved around the area, but larger objects like a keyboard and monitor are stationary.

Currently, our technique can account for one major source of environment illumination, such as a single lamp or window. The result is much less effective when there are multiple significantly bright light sources, especially when they are spaced far apart on the environment map (as the illumination becomes more homogeneous). In these situations, the result is equivalent to normal OpenGL blending.

Our post-processing exposure adjustment operates on an assumption about the distribution of actual intensities

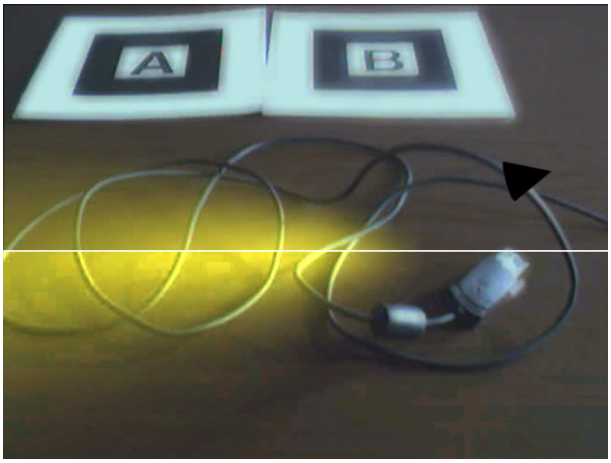

Figure 8: A physical table and wire illuminated with a virtual spotlight coincident with the physical illumination. The top half is using alpha blending, while the bottom half is using alpha blending. Note tha with our shader, highlights are brighter and shadows are darker.

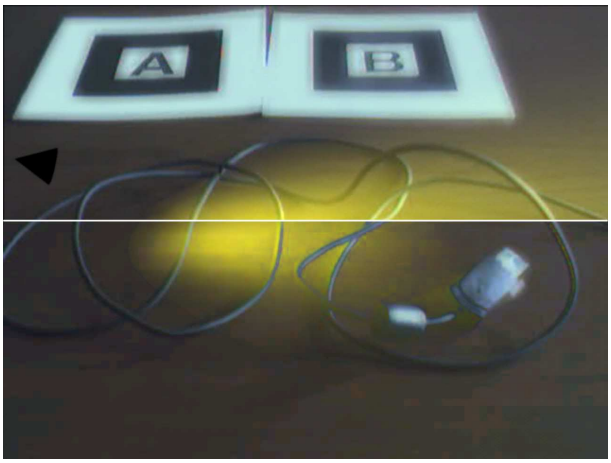

Figure 9: A physical table and wire illuminated with a virtual spotlight opposing the physical illumination. The top half is using alpha blending, while the bottom half is using our shader. Note that with our shader, highlights stay white and shadows become illuminated.

of saturated pixels in an image. This assumption more accurately models the reality in some cases than others. Additionally, since the darkened pixels are all originally saturated, more information about their intensities after darkening cannot be generated – uniformly gray halos will appear for heavy darkening. Bright illumination also emphasizes the inaccuracies of the light model, making them more obvious. For subtle lighting effects with soft boundaries however, the particular insensitivities of the visual system make these artifacts much less noticeable [8].

5 Conclusions and Future Work

In this paper, we present a system which addresses a variety of issues in the realm of consistent illumination. The main contributions are a means of acquiring and rendering with physical environment illumination, and a method to integrate virtual illumination effects with physical geometry, all in real-time with minimal setup cost. The first contribution is not a novel algorithm, but the application of disparate techniques to a common problem to create a single real-time system with dynamic response. The second contribution is a pair of techniques that allow for effective approximation of virtual lighting in complex and dynamic physical scenes. Combined, this represents a step forward in consistent illumination for augmented reality systems.

Our system would clearly be improved by the incorporation of virtual shadowing of physical geometry, possibly through similar means as the virtual illumination. Differential rendering can also subtract illumination from an image, which would be necessary for shadow generation. The main obstacle to shadow generation is a reasonable means of generating approximate soft shadows from environmental lighting stored in an environment map.

More accurate rendering of environment illumination on virtual objects, accounting for shadows from physical objects affecting virtual rendering, would also significantly add to the perceived realism. Rough physical shadow volumes could be generated from the simplified scene geometry and environmental lighting data, which could then be used in the traditional manner when rendering the virtual geometry.

The biggest hardware based improvement to our system would come from a high dynamic range camera. As in most image based lighting systems, high dynamic range imagery drastically improves the realism of rendered images [6]. Techniques are being developed to capture high dynamic range image data in real-time [22, 14], and graphics hardware is capable of rendering with high dynamic range image data, so the addition of this technique is feasible in real-time and would create much more

realistic material reflectances.

Acknowledgements

This work is supported in part by NSF IGERT: Graduate Training Program in Interactive Digital Media, award number DGE-0221713, and an equipment donation from Microsoft.

References

[1] Kusuma Agusanto, Li Li, Zhu Chuangui, and Ng Wan Sing. Photorealistic rendering for augmented reality using environment illumination. In *ISMAR*, October 2003.

[2] Michael Ashikhmin and Abhijeet Ghosh. Simple blurry reflections with environment maps, 2002.

[3] Oliver Bimber, Anselm Grundhöfer, Gordon Wetzstein, and Sebastian Knödel. Consistent illumination within optical see-through augmented environments. In *ISMAR*, pages 198–207, October 2003.

[4] James Blinn and Martin Newell. Texture and reflection in computer generated images. In *Communications of the ACM*, pages 542–546, October 1976.

[5] Brian Cabral, Marc Olano, and Philip Nemec. Reflection space image based rendering. In *SIGGRAPH*, pages 165–170, August 1999.

[6] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH*, July 1998.

[7] Stephen DiVerdi, Daniel Nurmi, and Tobias Höllerer. ARWin - a desktop augmented reality window manager. In *ISMAR*, October 2003.

[8] James Ferwerda, Sumanta Pattanaik, Peter Shirley, and Donald Greenberg. A model of visual masking for computer graphics. In *SIGGRAPH*, pages 143–152, August 1997.

[9] Alan Fournier, Atjeng S. Gunawan, and Chris Romanzin. Common illumination between real and computer generated scenes. In *Proceedings of Graphics Interface*, pages 254–262, 1993.

[10] Simon Gibson and Alan Murta. Interactive rendering with real-world illumination. In *Proceedings of 11th Eurographics Workshop on Rendering*, pages 365–376, 2000.

[11] Masayuki Kanbara and Naokazu Yokoya. Geometric and photometric registration for real-time augmented reality. In *ISMAR*, 2002.

[12] Jan Kautz and Michael McCool. Approximation of glossy reflection with prefiltered environment maps. In *Graphics Interface*, pages 119–126, 2000.

[13] Céline Loscos, George Drettakis, and Luc Robert. Interactive virtual relighting of real scenes. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):289–305, 2000.

[14] Shree Nayar and Tomoo Mitsunaga. High dynamic range imaging: Spatially varying pixel exposures. In *Computer Vision and Pattern Recognition*, 2000.

[15] NVIDIA. Cg: C for Graphics. http://www.nvidia.org/.

[16] Ivan Poupyrev, Desney Tan, Mark Billinghurst, Hirokazu Kato, Holger Regenbrecht, and Nobuji Tetsutani. Developing a generic augmented-reality interface. *Computer*, 35(3):44–50, March 2002.

[17] Ravi Ramamoorthi and Pat Hanrahan. Frequency space environment map rendering. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, July 2002.

[18] Ramesh Raskar, Greg Welch, and Kok lim Low Deepak Bandyopadhyay. Shader lamps: Animating real objects with image-based illumination. In *Proceedings of the 12th Eurographics Workshop on Rendering*, June 2001.

[19] Mel Slater, Martin Usoh, and Yiorgos Chrysanthou. The influence of dynamic shadows on presence in immersive virtual environments. In *Proceedings of the 2nd Eurographics Workshop on Virtual Environments*, pages 8–21, January 1995.

[20] Andrei State, Gentaro Hirota, David Chen, William Garrett, and Mark Livingston. Superior augmented-reality registration by integrating landmark tracking and magnetic tracking. In *SIGGRAPH*, pages 429–438, August 1996.

[21] Natsuki Sugano, Hirokazu Kato, and Keihachiro Tachibana. The effects of shadow representation of virtual objects in augmented reality. In *ISMAR*, pages 76–83, October 2003.

[22] Jamie Waese and Paul Debevec. A real time high dynamic range light probe. In *SIGGRAPH Technical Sketch*, 2001.

[23] Robert Welch, Theodore Blackmon, Andrew Liu, Barbra Mellers, and Lawrence Stark. The effects of pictorial realism, delay of visual feedback, and observer interactivity on the subject sense of presence. In *Presence: Teleoperators and Virtual Environments*, volume 5, pages 263–273, 1996.

[24] Lance Williams. Casting curved shadows on curved surfaces. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 270–274, 1978.

[25] Bob Witmer and Michael Singer. Measuring presence in virtual environments: A presence questionnaire. volume 7, pages 225–240, 1998.