Synthesizing Environment Maps from a Single Image

Jean-François Lalonde and Alexei A. Efros
{jlalonde,efros}@cs.cmu.edu

CMU-RI-TR-10-24

Originally developed in 2008
Published in July 2010

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Environment mapping is a popular technique for creating consistent lighting when compositing a virtual object into a real scene. However, capturing an environment map usually requires physical access to the scene to obtain illumination measurements. But what if all one has available is a single photograph of the scene? In this paper, we study techniques for synthesizing *plausible* environment maps from a single image. By analogy with texture synthesis, the goal is to use the small amount of available data to generate an environment map that would be likely to have come from that scene. In particular, we are interested in understanding the role of geometric information in constructing visually realistic environment maps. To this end, we implement several environment synthesis strategies that employ varying amounts of 3D scene geometry information. We measure the quality of the synthesized results by using human subjects to evaluate the appearance of objects illuminated with different environment maps, in still images as well as in video.

Contents

Figure 1: Our method generates plausible location-dependent environment maps from a single image, and enables the insertion of 3-D objects into photographs to generate realistic renderings and animations.

1 Introduction

In many image and video editing applications, it is often desirable to seamlessly place a virtual object into a real scene. Such applications range from adding virtual spaceships into a sci-fi movie to rendering a sofa into your living-room to see how it matches the rest of the furniture. To make sure that the synthetic object "fits" into the scene, it is important to match object properties with those of the target image. Parameters such as object position, scale, and orientation need to be carefully chosen. However, in practice, even if all of these are accurately matched, the synthetic object will still most likely look out of place, often perceived as "floating" above the scene. The main reason for this has to do with lighting and shadows. The appearance of an object in a real scene is affected by light from its entire environment: sky, trees, grass, buildings, etc. In comparison, a synthetic object is usually lit in a boring, simplistic way, making it immediately stand out when composited into a real scene.

The way this problem is usually addressed is by capturing natural light from the real scene and using it as an environment map to light the virtual object [7]. An environment map [3] is a sample of the plenoptic function [1] at a single

point in space capturing the full sphere of light rays incident at that point. It is typically acquired by either taking a panoramic photograph from the point of view of the object, or by placing a mirrored ball at the desired location and photographing it. Such environment map can then be used as an extended light source for rendering the synthetic object.

Capturing environment maps this way assumes that one has physical access to the background scene, allowing measurements to be taken. But what if all you have is a *single image*? For example, a historical photograph, a painting, or just some image off the web. Is there a way to obtain an environment map from just that information alone? Strictly speaking, the answer is no. Of course, if there just happens to be a mirrored sphere in the image or some other reflective object of known geometry this is indeed possible (e.g. see [17] for a very inventive use of the human eye). However, for a general scene, the only valid environment sample in the image would be the pixel along the ray from the viewer to the desired center of the environment map.

But do we actually need a perfect environment map to light an object in a believable way? It is well-known that painters can often get away with wildly inconsistent lighting [6]. This is because humans are extremely poor at reasoning about the physics of light, in particular, illumination and reflections [2]. Moreover, Dror et al. [8] have shown that real-world illumination possesses a high degree of statistical regularity. They demonstrated that while random or highly unlikely illumination does not look natural, environment maps created using a simple texture synthesis approach [19] produced believable lighting for non-mirrorlike objects. This all suggests that a plausible environment map which is consistent with the target image may be sufficient for creating realistic lighting effects, even for more mirror-like objects.

Indeed, digital artists have long been known to "fake" environment maps. Typically this is done by mapping the background image onto a cylinder and applying either stretching or mirroring to produce a full 360° map (e.g. see online tutorials [4, 5]). Last year, Khan et al. [14] proposed a related approach, projecting a circle in the image onto a sphere and mirroring. In either case, to make sure that the resulting environment maps have high dynamic range, one usually either starts out with an HDR image, or produces one by hand by eyeballing a reasonable gamma correction.

Another popular graphics trick is even simpler: instead of synthesizing a new environment map, one simply uses an existing light-probe image, even if it doesn't really match the background scene (see [8] for evidence that this often works). Other approaches to object relighting aim to avoid environment map construction altogether. Reinhard et al. [20] show that object recoloring after rendering, based on the colors of the background image, is often quite effective. Meanwhile, Lalonde et al [15] take a data-driven approach, choosing among a library of objects ones that have been captured under similar lighting conditions.

It is unfortunate, however, that despite the plethora of approaches, we are not aware of any attempts to compare them in a systematic way. Moreover, a major shortcoming of all environment map estimation approaches is that, given an image, they produce a single environment map, regardless of where the

virtual object is to be placed in the scene. This is particularly problematic if we wish to have the object move around within the image. Not only will the estimated lighting not take into account the particular position that the object is in, but the fact the the lighting stays constant while the object is moving will look highly unrealistic in many cases.

2 Overview

To address the issues raised above, we propose a new method for synthesizing plausible, location-dependent environment maps from a single outdoor image. Our overall philosophy is to first account for all the light that has been captured in the image, and then use that to help us synthesize the rest of the environment map (note that in order to use any of the pixels in the image, we will have to assume that the scene is mostly Lambertian, i.e. each surface point emits the same amount of light in all directions). Given an arbitrary 3D point in the depicted scene, our aim is to construct a full spherical environment map around it.

The problem is that without any notion of the scene geometry it is impossible to know how much of the point's visual field the image occupies. For example, an image of a narrow alleyway will occupy more than half of the environment map for a point inside the alley. But an image of an open field would occupy only a small fraction of the point's environment. Fortunately, for this task we only need a very rough idea of the geometry – just enough to disambiguate coarse scene structure. Such information is obtainable, even from a single image, either automatically [12] or using one of the user-guided single-view modeling systems [13, 18]. Once rough geometry is known, it is possible to reasonably accurately fill in the environment map with known values from the image.

Depending on the scene and the chosen scene point, the above approach will estimate from a third to more than half of the environment map values. Unfortunately, most of it is on the "wrong" side of the environment sphere – toward the image and away from the viewer. While this information is valuable (e.g. for rendering transparency), it will be of little use for compositing most virtual objects. What we need is to estimate scene appearance from the side of the viewer – something for which there is absolutely no data! All we know is that, statistically, the unseen side should not be too different from the observed one [8]. The good news is that the observer does not know the right answer either, so anything plausible would do. Here our solution is similar to the previous approaches – we assume that the unknown side of the environment map is just a mirror copy of the known side. This not only encodes the symmetry that is often present in man-made architectural scenes, but also plays to human tendency to expect symmetry in the world. Note that we are not guaranteed to be able to fill in the entire backside hemisphere. Therefore mirroring might still leave unfilled holes in the environment map. We solve this problem by applying further local mirroring and/or texture synthesis within each major estimated geometrical region. While this approach is simple, it tends to produce surprisingly

good quality environment maps, usually without any visual artifacts.

The rest of this paper is divided into two parts. In the first part (Section 3), we describe the details of our geometry-based approach to location-dependent environment map synthesis. Because it is not clear *a priori* how detailed the scene geometry information needs to be, we will propose a series of progressively more complex synthesis techniques. In the second part (Section 4), the above approaches will be evaluated in a user study, and compared to other techniques as well as to ground truth where available. Evaluation will be done on still images, as well as video clips with moving objects.

3 Synthesizing Location-dependent Environment Maps

Given the input image and a desired (x, y, z) location, our goal is to synthesize a *plausible* environment map centered at that location by using the available image data. We first describe how we estimate rough scene geometry from a single image. We will then explore several environment map synthesis strategies that employ varying amounts of 3-D scene geometry information. Finally, we will discuss the issues related to the dynamic range of the resulting environment maps.

3.1 Estimating Scene Geometry

Given an image, we would like to estimate the basic scene geometry (major planar surfaces in the scene), the horizon position, the camera focal length, and the camera height. Note that most of this information will be required anyway to align the virtual camera with the real one for object insertion. We borrow the definition of rough scene geometry from recent single-view reconstruction papers [13, 12] which minimally consists of a set of polylines denoting the ground-vertical boundaries. These polylines and the horizon position can be obtained either by user input or automatically, using Automatic Photo Pop-up approach [12]. The camera focal length can be read from the image meta-data or, if unavailable, set to a reasonable value (we use $1.4 \times (\textit{image width})$ as default, typical for a 35mm camera). The camera height is required to determine the correct size of synthetic objects to be inserted. In practice, the camera height can be defaulted to 1.6 meters (average eye level) or manually estimated using the heights of known objects in the scene.

3.2 Spherical geometry

Let us start by ignoring the available geometric information and making the assumption that the scene geometry can be roughly approximated by a sphere centered in the middle of the scene. This is the method proposed by Khan et al. [14]. They carve out the largest possible circular part of the image and map it onto a hemisphere behind the image, and a hemisphere in front of the image.

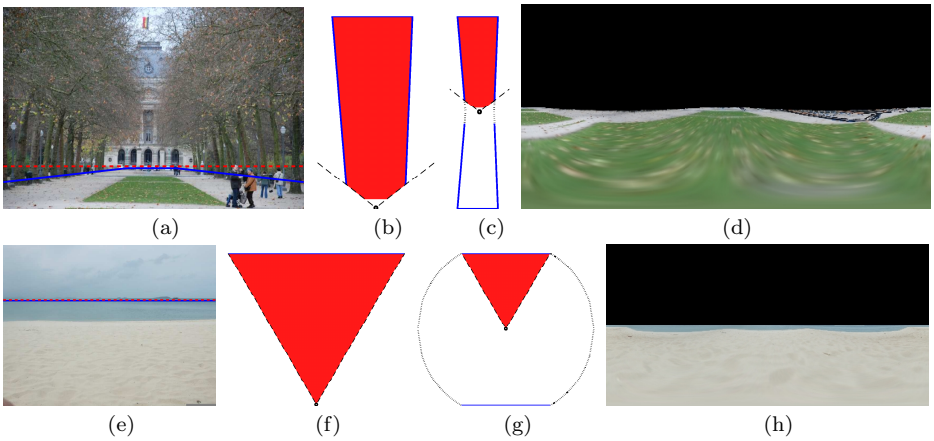


Figure 2: Geometry extrapolation in a perspective (top) and open (bottom) scenes. From the input image and its rough geometry labels (a,e), we back-project the labels onto the $y = 0$ plane (b,f) and extrapolate to get a ground plane polygon (c,g). Projecting the reflections back into the image and interpolating to get color values yields the resulting environment map (d,h). Shown in blue are the geometry labels, the dotted red line is the horizon line and the red polygon is the visible portion of the ground.

Throughout this paper, we will call this technique `Spherical`. Figure 3(b,e) show examples of this mapping. Note that, as described, this technique is not location-dependent, making comparisons with other approaches somewhat unfair. However, since it assumes a spherical scene geometry, it is not difficult to render location-dependent environment maps by projecting from the big sphere centered at $(0, 0, 0)$ onto smaller spheres centered at (x, y, z).

3.3 Ground+Hemisphere geometry

Let us now add one piece of scene geometry by forcing the ground to be flat in the resulting environment map. This effectively approximates the environment by a plane and a top hemisphere. By using the approximated camera parameters, we can back-project the ground-vertical polylines on the 3-D ground plane using the following approach, inspired by [12].

Under a perspective camera model, assuming a well-behaved camera (unit aspect ratio, zero skew, no distortion), we can convert between image and 3-D coordinates using the following equations:

$$u_i = x_i \frac{f}{z_i} + u_0 \tag{1}$$

$$v_i = (y_c - y_i) \frac{f}{z_i} + v_0 \tag{2}$$

where (u_i, v_i) are the image coordinates of point i, f is the camera focal length, (x_i, y_i, z_i) are the 3-D coordinates (shift, height, depth), u_0 is the optical center

(typically half of the image width), v_0 is the horizon position in the image, and h_c is the camera height. These equations assume zero camera tilt for simplicity of discussion (removing this assumption would merely require specifying the horizon line instead of the position v_0).

Setting the ground plane as $y = 0$, we can determine the 3-D coordinates of the visible portion of the ground based on the geometry labels and the image edges, as shown in a top-down view on Figure 2(b,f). Unfortunately, a (potentially large) portion of the ground is invisible to the camera, so we need to extrapolate this unseen geometry and appearance.

To extrapolate the geometry, we extend the lines intersecting the image edges until they reach the $z = 0$ line. There are two cases to consider: 1) a road or alley going into the distance, where the ground boundaries should be extended straight toward the viewer (Figure 2(top)) and 2) an open scene with just the horizontal ground boundary, which should be smoothly curved inward to avoid infinitely wide scenes (Figure 2(bottom)). We use spline interpolation to smoothly blend between straight lines and curved extrapolations, which handles both of these scenarios automatically. Finally, to determine the geometry behind the camera, we simply reflect about the camera plane.

Having extrapolated the ground polygon on the $y = 0$ plane, we now generate an environment map using the available image information. For each 3-D direction that intersects the ground polygon, we first determine if it falls on the visible region (or its reflection), in which case we project the intersection point back into the image and determine the its color via bilinear interpolation. For the directions that intersect outside the visible region, we try to reflect them along the z-coordinate of the visible region. If a reflected intersection still lies outside the visible region, then the intersection is reflected about the nearest visible region segment. After this procedure, all reflected points which fall into the visible region are projected in the image and bilinear interpolation is performed to retrieve the color values. Nearest-neighbor interpolation is done for the points which still do not reflect inside the visible region.

We use the top part of the hemisphere from the `Spherical` technique to fill-in the top half of the environment map by aligning the horizon to avoid strong discontinuities. We term this technique: `Ground+Hemisphere`, with an example shown on Figure 3(c). Note that since all the necessary computations are simple 3-D operations (projections and reflections), this technique can be computed very efficiently.

3.4 Ground+Vertical+Sky

Now let us take a step further and assume that the scene consists of a single ground plane, planar surfaces perpendicular to that plane, and the top hemisphere representing the sky ([12] show that this assumption leads to qualitatively valid models for many scenes).

To build the surrounding vertical walls into our representation, we follow an approach very similar to the one introduced in the previous section, but we use vertical planes instead of a horizontal one. We define one plane for each

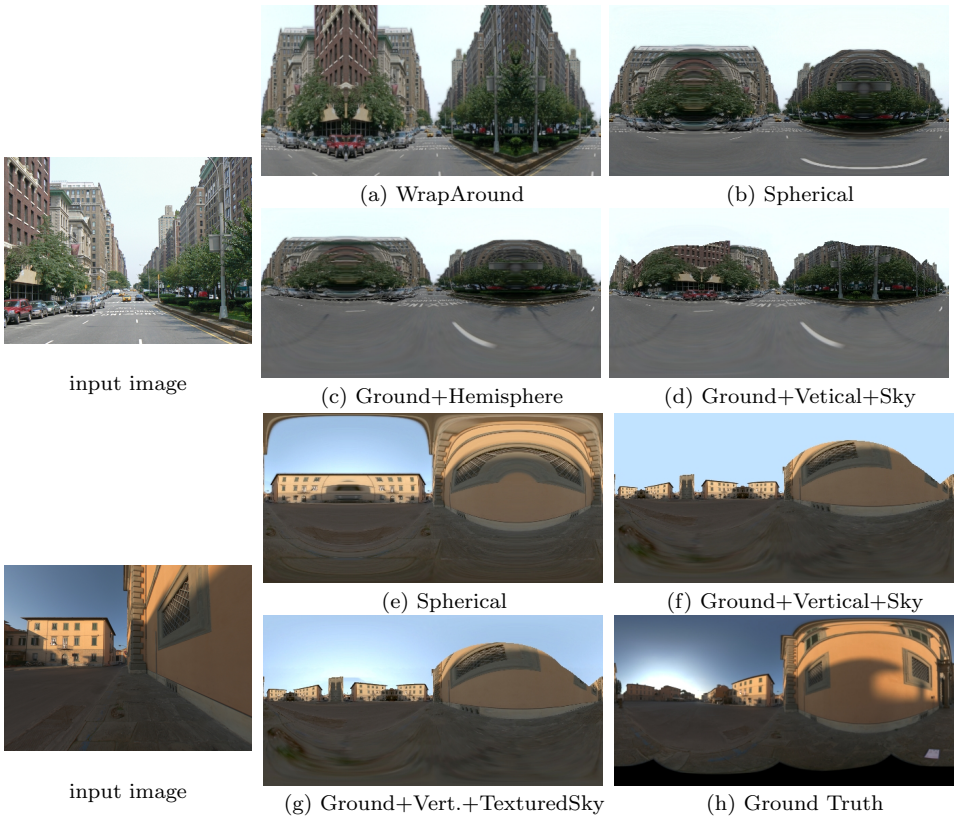


Figure 3: Different ways of synthesizing environment maps from a single image. From an input image (a), environment maps (shown here in latitude-longitude format) are synthesized using the techniques described in the paper.

edge of the ground polygon. To determine the height of the visible walls, we assume that all pixels that lie above a pixel on the ground/vertical boundary have the same depth. Given the known depths, we can compute the height of each wall by calculating the height of the highest pixel in the image using equations 1 and 2, at each of its ground vertices. We extrapolate the wall height by keeping them constant. Reflection, re-projection and bilinear interpolation are then performed in the same way as in the previous section to recover the color for each direction in the environment map. For the top hemisphere, we simply fill it with a constant sky color sampled from the sky region. We refer to this technique as `Ground+Vertical+Sky`, and example environment maps obtained with this technique are shown in Figure 3(d,f).

3.4.1 Textured Sky

The previous method used a constant average color to fill-in the missing portions of the sky, but this lack of texture often results in dull-looking composites. To alleviate this problem, we need to fill-in the top hemisphere with the available sky information from the image.

To do so, we use [12] to compute a sky probability map from the image and extract a sky mask. We then compute the environment map representation of the mask by applying the `Ground+Vertical+Sky` technique, which tells us what part of the environment map we just computed is actually sky, not walls. Finally, texture synthesis [9] is used to fill-in the sky region, using the sky mask as source. Please note that texture synthesis is performed in the angular representation to avoid the stretching effect at the poles inherent to the latitude-longitude representation (see [21] for more details about various environment map representations and their properties). We term this technique `Ground+Vertical+TexturedSky`. See Figure 3(g) for an example.

3.5 WrapAround baseline

Finally, let us define a simple baseline technique that is commonly used in practice. The input image is first mirrored in the x direction and then simply wrapped around a sphere by assuming that it is in the latitude-longitude format. See Figure 3(a) for an example result.

3.6 Dynamic Range Extrapolation

Most photographs taken today have a low dynamic range and are not photometrically calibrated. This is an important issue since, for correct rendering, environment maps must contain true radiance values (up to a scale factor). We partially alleviate this problem by first estimating an approximate camera response function, and artificially increasing the radiance of saturated pixels.

3.6.1 Camera response function

To recover the true radiance captured by the camera (up to a scale factor), one would need its response function. Since we are not making precise light measurements, we do not require the exact response function, and an approximation is sufficient. Several options are possible: estimate the camera response function from a single image [16]; use an average response function as in [11]; or even use only a fixed gamma function [22] for all images.

We choose to employ the simple frequency-based technique proposed by Farid [10] and estimate a value for gamma, which is then used to linearize the image. This technique reportedly yields estimates within 7.5% of their true value on average.

3.6.2 Saturated pixels

Applying the inverse gamma function attempts to linearize the pixel values to scaled radiance units in the areas of the image that are correctly exposed. However, in over-exposed regions, the recovered value does not faithfully represent the scene radiance since the camera pixel was saturated. Since the scene radiance of saturated pixels should proportionally larger than the correctly exposed areas, we follow the approach proposed by [22] and artificially increase the saturated pixel radiance. This is done by computing saturated pixels mask in any of the three channels (pixels values of 254/255 and above are considered saturated) and blurring it with a large-bandwidth gaussian kernel. This resulting mask can then be used to scale the saturated pixels variance by a fixed factor (we used 3 times the brightest non-saturated pixel in the image).

3.6.3 The Sun

Consider the renderings shown in the first row of Figure 4, in which the rendered objects both have a very unnatural blueish tint. This is caused by the fact that the environment maps used to light the objects were obtained by extrapolating image information, but the image itself does not contain the most important light source: the sun! Unfortunately, automatically estimating the sun direction and intensity from a single image is a very challenging computer vision problem, so we bypass it for the moment by allowing the user to pick these parameters. Since photographers tend to shoot with the sun in their backs, placing the sun behind the camera is a good default value, and was used in the examples of Figure 4.

4 Evaluation

In the previous section we have developed a number of techniques for using rough scene geometry to generate position-dependent environment maps from an image. How can we tell which will work best in practice? We could try to compare the generated environment maps to each other and against the ground truth using some kind of an error metric. However, studies have shown that lighting is largely a low-frequency phenomenon and that humans are anyway quite poor at telling whether objects are lit in a physically correct way [8, 6]. Therefore, instead of evaluating the quality of the environment maps directly, we will consider how well they perform at realistically lighting virtual objects in real photographs. Since the question of realism is inherently subjective, relying on the judgment of human observers, the evaluation was performed via a user study. In this section, we will first describe how we generated virtual object renderings. Then we will describe our two user experiments: on still images and on video sequences.

Figure 4: Effect of adding a synthetic sun to the environment map. *First row*: objects rendered using our automatically-generated environment maps. *Second row*: same objects rendered using the same environment maps augmented with an additional light source acting as the sun. The sun's brightness was set to be 1×10^4 times brighter than the maximum image brightness.

4.1 Object Insertion

Using the estimated environment maps, we are able to render virtual objects at specified locations in the real scenes. We use the estimated scene geometry to setup the virtual camera parameters for our renderer. To obtain realistic shadows from the virtual object on the real ground, we follow [7] modeling part of the ground plane and use subtraction to get the correct shadows. We used the RADIANCE software package for all our renderings. Some examples, using the `Ground+Vertical+Sky` technique are shown on Figure 5.

4.2 Still Image User Study

For this experiment we have created 12 different composite scenes, combining various background images (outdoors) with a set of 3D virtual object models (both shiny and matte). For each scene, the object has been rendered five times, using environment maps estimated with the techniques described in the previous section: `WrapAround, Spherical, Ground+Hemisphere, Ground+Vertical+Sky, Ground+Vertical+TexturedSky`. Additionally, as is sometimes done in prac-

(a) (b) (c)

Figure 5: From a single input image and rough geometry, we extrapolate the scene geometry and the dynamic range of the image, allowing the creation of an environment map from any point in the scene. Here we show various input images (a), their synthesized environment maps (b) and inserted objects (c). Rows 1 and 2 were generated using automatic geometry estimation [12]; rows 3 was generated from user-provided labels.

tice, we have also rendered each object with an existing light-probe environment map (we used Debevec's popular eucalyptus grove map). We also implemented an image-space object insertion algorithm of Reinhard et al [20], which recolors the image of the inserted object based on the colors of the background image. The object still needs to be lit somehow before it could be recolored. Instead of a simplistic point-light rendering, we opted for, again, using a standard light-probe image. Finally, 7 of the backgrounds in our test set have been created by "virtually photographing" part of a light-probe environment maps (taken from Debevec's extensive online light-probe collection). Therefore, for these scenes, we actually have the full ground truth environment map available, so we can render the object in the physically correct way. Some of the example renderings are shown on Figure 6.

In summary, for each of the 12 composite scenes, we have either 7 or 8 (if ground truth is available) renderings. For each of the 15 subjects in our user

study, we have presented them with all rendered versions of each scene and asked to rank them, from most realistic to least realistic. All 7 or 8 images were presented on the screen at once, but the users were allowed to zoom in, if desired. They were required to produce a strict ranking, no ties. No time limit was set.

The results of the study turned out to be quite interesting, as shown on Figure 7. The first thing to notice is that there is no clear winner. While the ground truth renderings are slightly ahead, it is not statistically significant. This reaffirmed the notion that human subjects are not necessarily interested in physically correct renderings. For example, Figure 9 shows an image where most of the subjects ranked the `Ground+Hemisphere` approach significantly higher than ground truth. Second, it appears that for still images, a simpler geometric model of the scene often works as well or better than a more complex one. The fact that `Ground+Hemisphere` and `Spherical` [14] perform better than the rest might suggest that humans also have a very simplified model of scene lighting, and so find these results most consistent with their internal representation. Finally, it appears that despite its popularity, the approach of using a standard environment map for relighting does not perform well in practice, even if followed by image-based recoloring. The only caveat is that digital artists probably pick a light-probe that they believe is most consistent with the image they are working with, whereas we, for consistency, have chosen a single popular light-probe for all images.

4.3 Animation User Study

The real advantage of location-dependent environment map synthesis is the ability to animate virtual objects in real scenes (see Figure 1 and supplemental video). However, a reasonable question to ask would be: how perceptually important is lighting consistency for a moving object? To answer this question, we have set up a second user study, asking subjects to rank the realism of video clips. Because video is inherently more time-consuming, we had to limit ourselves to 4 video clips and two techniques. Originally, we planned to compare a stationary (constant) environment map vs. a location-dependent one. However, in preliminary tests, the stationary map looked so unrealistic, that we decided to go for a more interesting comparison. The two techniques that we picked are our improved, location-dependent version of `Spherical` [14] and `Ground+Vertical+Sky`. The subjects would view a pair of video clips, one after the other, and then rank which one looked more realistic.

The results of the study are shown on Figure 8. By a very significant margin, the geometry-aware lighting was seen to be more realistic than the spherical assumption, even though both lighting strategies were location-dependent. The fact that there is such a striking difference in perception of lighting between the still vs. moving stimuli suggests that very different visual mechanisms might be in play in the two cases. Not only does it appear that scene geometry plays an important role in relighting moving virtual objects, but this finding suggests that more work needs to be done in the domain of virtual object animation, a

Figure 6: Comparison of different approaches for virtual object insertion.

topic that has yet to receive much attention.

5 Conclusion

This paper contains two major contributions. First, we have presented a set of techniques for generating location-dependent environment maps from a single, low-dynamic-range image. The main insight is to estimate rough geometric structure of the scene, which in turn allows us to synthesize environment maps that are more realistic and more consistent with the input image. Our second contribution is a user study comparison of several environment map synthesis approaches, which aimed to measure the importance of scene geometry on perception of realistic lighting. Our main conclusion is that while for still images the role of scene geometry is not very significant, its importance in object animation is very substantial.

References

[1] Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing*, 1991.

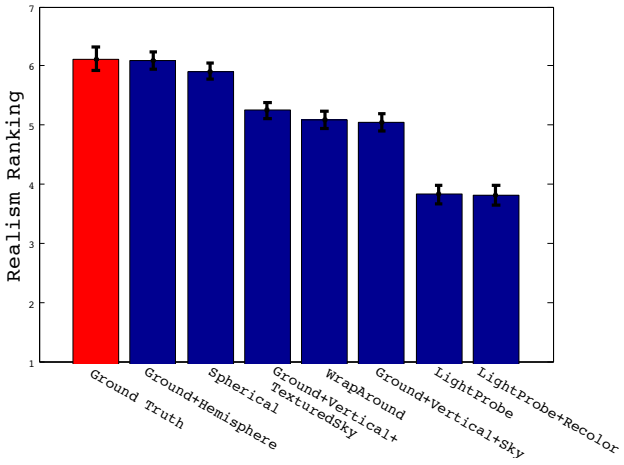


Figure 7: Human rankings of still images rendered with different techniques. Higher rank indicates more percieved realism. Ranking of renderings using the ground truth environment map is shown in red. It appears that for still images, no one technique is a clear winner, not even the ground truth.

[2] M. Bertamini, A. Spooner, and H. Hecht. Naive optics: Predicting and perceiving reflections in mirrors. *Journal of Experimental Psychology*, 29(5):982–1002, 2003.

[3] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, October 1976.

[4] Daniel Broadway. Image-based lighting tutorial. website. http://www.geocities.com/night_sky_vfx/IBL.html.

[5] Christopher Cameron. Hallucinating environment maps from single images. website, 2005. http://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/f05/pub/www/projects/fproj/cmcamero/.

[6] Patrick Cavanagh. The artist as neuroscientist. *Nature*, 434:301–307, 2005.

[7] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of ACM SIGGRAPH 1998*, 1998.

[8] Ron O. Dror, Alan S. Willsky, and Edward H. Adelson. Statistical characterization of real-world illumination. *Journal of Vision*, 4:821–837, 2004.

[9] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, August 2001.

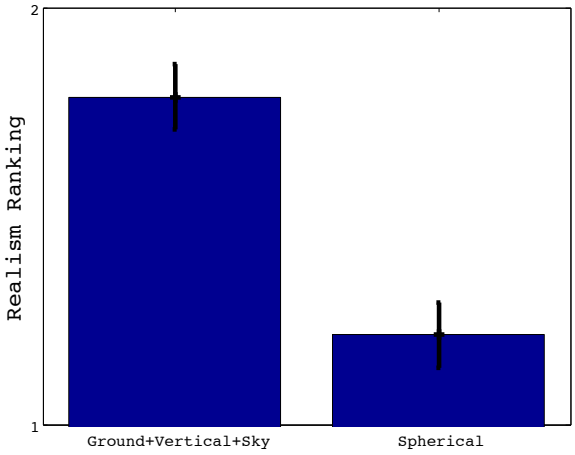


Figure 8: Human rankings of animation clips rendered using `GroundVerticalSky` and location-dependent version of `Spherical`. More geometrically correct lighting is strongly prefered by people over spherical geometry assumption.

[10] H. Farid. Blind inverse gamma correction. *IEEE Transactions on Image Processing*, 10(10):1428–1433, 2001.

[11] M. D. Grossberg and Shree K. Nayar. What is the space of camera response functions? In *IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

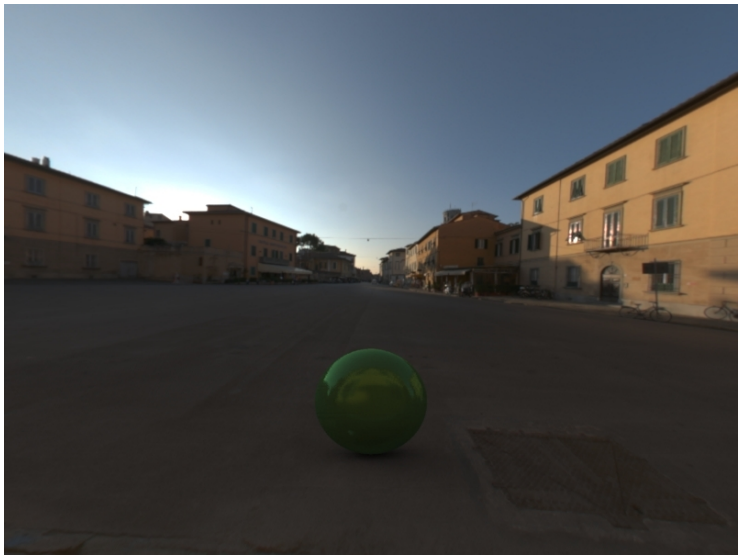
[12] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *ACM Transactions on Graphics (SIGGRAPH 2005)*, 24(3), August 2005.

[13] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *Proceedings of ACM SIGGRAPH 1997*, 1997.

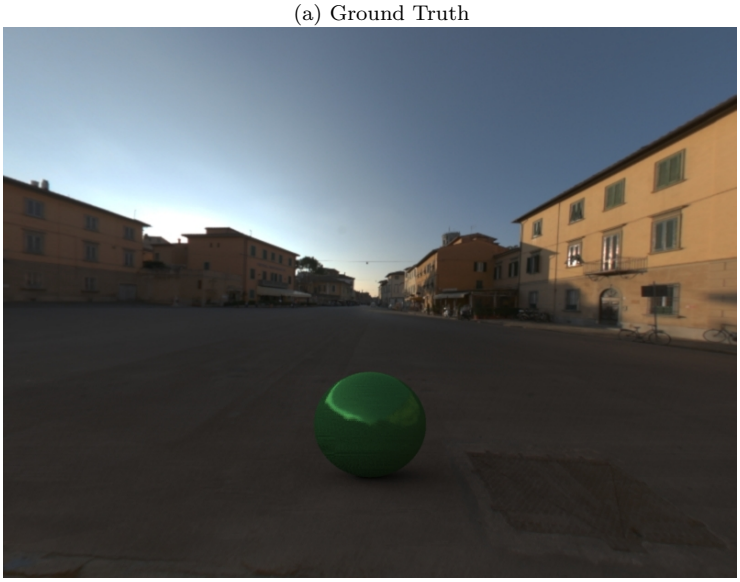
[14] Erum Arif Khan, Erik Reiard, Roland Fleming, and Heinrich Büelthoff. Image-based material editing. *ACM Transactions on Graphics (SIGGRAPH 2006)*, August 2006.

[15] Jean-François Lalonde, Derek Hoiem, Alexei A. Efros, Carsten Rother, John Winn, and Antonio Criminisi. Photo clip art. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), August 2007.

[16] Steve Lin Lin, Jinwei Gu, Shuntaro Yamazaki, and Heung-Yeung Shum. Radiometric calibration from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2004.



(a) Ground Truth



(b) `Ground+Hemisphere`

Figure 9: For this image, users consistantly ranked the synthetic lighting (b) higher than the ground truth lighting (a)

[17] Ko Nishino and Shree K. Nayar. The world in an eye. In *IEEE Conference on Computer Vision and Pattern Recognition*, July 2004.

[18] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *Proceedings of ACM SIGGRAPH 2001*, 2001.

[19] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal on Computer Vision*, 40(1):49–71, December 2000.

[20] Erik Reinhard, Ahmet Oguz Aküyz, Mark Colbert, Charles E Hughes, and Matthew O'Connor. Real-time color blending of rendered and captured video. In *Interservice/Industry Training, Simulation and Education Conference*, December 2004.

[21] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High dynamic range imaging*. Morgan Kaufman, 2005.

[22] Allan G. Rempel, Matthew Trentacoste, Helge Seetzen, H. David Young, Wolfgang Heidrich, Lorne Whitehead, and Greg Ward. Ldr2hdr: On-the-fly reverse tone mapping of legacy video and photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), August 2007.