Environment Matting Revisited

Figure 1: *A transparent wine glass, a colored dragon, a mirrored table, and a chess pawn have been digitally composited in novel environments. Myriad light transport effects such as refraction, reflection and selective attenuation have been preserved. Best viewed in color.*

Abstract

The most convincing environment matting (EM) techniques use a large number of (monochrome, or two-tone) probing images to extract the matte. In this paper, we use *multiple colors* as cues, noting that color encodes several bits of information. Compared to earlier works, we use substantially smaller number of preprocessed images in the form of a color cube environment. If c colors are used for the cube each of whose faces is $k \times k$, the number of images we needed is $\lceil \log_c 6k^2 \rceil$. The number six comes in becomes we want to emphasize that the problem needs to be solved in a complete environment. Further, we also provide a version in which only 1 image is used for real-time imaging purposes.

We show quantitative as well qualitative results in the form of images that exhibit sophisticated illumination effects. The rendered objects show effects such as highlights, simultaneous refraction and reflection (instead of pure specular refraction). Objects my be colored, and influence the attenuation of light. We have minimal memory and computational requirements. Applications of this work include the relighting for virtual and augmented reality.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation: — Display algorithms ; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism: — Color, shading, shadowing, and texture.

Keywords: image-based rendering, environment matte, colored structured patterns, cube-map, refraction, reflection, colored transparency, real-time capture.

1 Introduction

It is fascinating to look at transparent objects exhibiting complex, yet beautiful, optical properties. The breathtaking beauty emanates from the effects of refraction and reflection, often coupled with glossy and light scattering effects. Further, transparent, colored materials exhibit selective wavelength-dependent attenuation resulting in nice color shifts. These effects are generated due to the interplay of the involved light matter interaction that occurs when light hits the boundaries of the transparent object or travels through it.

Techniques based solely on captured images, which can be used to change the illumination of a scene or of an object, are termed as image-based relighting (IBRL) techniques. Lighting can be changed by introducing new light sources in a scene, modifying the existing light sources (position, intensity) or changing the environment in which the object is placed. Traditional matting and compositing techniques can be interpreted as IBRL methods. It involves capturing an image of an arbitrary object in an environment and then determining the color and opacity at each image pixel, represented by an *alpha channel* [Porter and Duff 1984]. Traditional compositing then simply involves placing (relighting) this object in a novel environment using the opacity to control the relative contributions to each pixel.

A typical matting and compositing technique, Blue screen matting [Smith and Blinn 1996], captures images of the object with tailored backdrops, but it fails to render the transparency correctly after matte extraction. Although traditional matting and compositing have proven tremendously useful in film, video, and computer graphics production, they nevertheless fail to simulate the key effects exhibited by transparent objects, that are essential for realism. To combat this, impressive methods have been developed in the recent past as mentioned in the next section.

1.1 Related Work

In [Zongker et al. 1999], the authors develop a mathematical framework for modeling the effects of refraction and reflection of light passing through transparent objects, by analyzing several captured images of the objects in front of hierarchical two-color patterned backdrops (and sidedrops). Once the model is created, the object can be composited in novel backgrounds, with all the relevant effects. But this model had several limitations, serious among them being that the contributions to a pixel of the foreground image was considered from only **one** region of a backdrop (sidedrop). In reality, multiple contributions (of regions) corresponding to the same backdrop are observed, for example when reflection and refraction

result in two groups of rays that strike the same backdrop. These issues were tackled in an extended and more accurate model of Environment Matting(EM) and compositing [Chuang et al. 2000] at the expense of using significantly *additional backdrops*. Nevertheless, as pointed in [Peers and Dutre 2003; Zhu and Yang 2004], problems remain. One issue is the error prone non-linear optimization which is mitigated by using a large number of "wavelet patterns" [Peers and Dutre 2003] or working in the frequency domain [Zhu and Yang 2004]. Another issue is moving from a parametric Gaussian framework to the use of the non-parametric framework [Wexler et al. 2002; Zhu and Yang 2004; Peers and Dutre 2003]. Notable in [Peers and Dutre 2003] is an EM extension where they simulate complex effects by capturing (basis) images of the object in front of thousands of wavelet illumination patterns, determined on the fly depending on the observed properties of the environment matte. During compositing, the novel background is first projected onto the space of those wavelet illumination patterns to obtain the appropriate coefficients. These coefficients are then used in conjunction with the basis images to compute the composite image. A major strength of this approach is that it captures diffuse reflections, which were found to be difficult to capture with the other approaches [Zongker et al. 1999; Chuang et al. 2000] since the illumination due to a sweep stripe was too weak. A possible solution, in that context, of acquiring high-dynamic images was proposed in [Chuang 2004]. [Matusik et al. 2002] combine EM [Chuang et al. 2000] and reflectance fields [Debevec et al. 2000] to acquire an image-based representation of transparent objects. They also reconstruct the 3D shape of the object, which allows the object to be viewed from novel viewpoints by interpolating between the environment mattes and reflectance images of nearby viewpoints.

In all of the above methods, the backdrop images were under user control. [Wexler et al. 2002] presented an EM extension that is able to work without the knowledge of the exact form of the backdrop images used. The method, however, relies on having enough background samples or sufficiently rich backdrop images (e.g. by moving a backdrop image behind the scene) to successfully extract an environment matte. Further, their method is effective for capturing colorless, perfectly specular objects (with a possible opaque alpha channel) only.

1.2 This Paper and Our Contributions

Despite the impressive achievements, the principal disadvantages of the above methods is the use of a large number of backdrop images. For example, [Peers and Dutre 2003] requires a huge storage for the basis images (average size of 2.5GB). The time limit for determining the wavelet illumination patterns and capturing the corresponding basis images was set to 12 hours. Similarly, [Zhu and Yang 2004] requires about 675 images for a 320x320 sized image.

- In general, all the methods which assume multiple regions of the backdrop can impact the transparent object require at least $2k \times 2k$ input images, for an output resolution of $k \times k$. Our method *uses a color space decomposition approach* that requires approximately $\log_c k$ input images where it is important to note that the base of the logarithm is under our control. This results follows because our backgrounds are coded with c-coded decimals instead of a binary coded decimal.

- In addition to modeling multiple regions from the same backdrop, we also model the fact that sidedrops and the floor, for example, can *simultaneously* impact the object (Figure 2). Therefore, unlike previous methods we embed the object in a complete cubic environment map (cube map). This enable us to discover any coupling between the various backgrounds. The idea here is to assign unique color codes over the entire cube map instead of a single face.

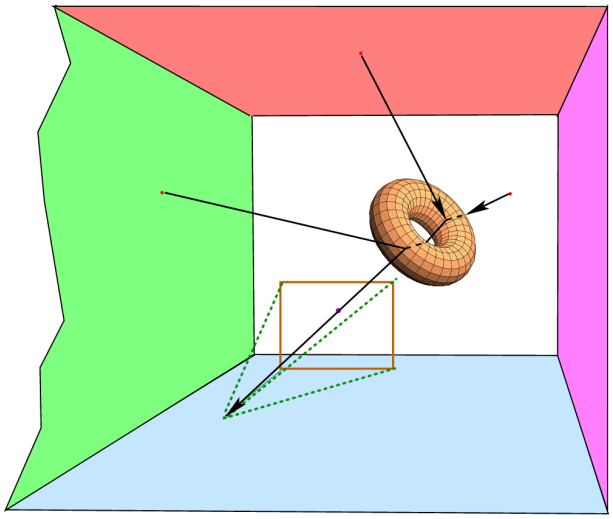


Figure 2: *In the environment matte (EM) problem, an optically active object (here, a torus) is captured (see view frustum) against various programmer defined backdrops. In this snapshot, the backdrop is solid white, and three pixels (the footprint) from the (red) ceiling, the backdrop, and the left (green) sidedrop contribute to the observed color. By analyzing a large number of backdrops and sidedrops, the torus can be digitally composited in novel environments. Our EM process uses a novel structured-colored cubic environment map instead of previous monochrome backdrops.*

- A notable exception to the large number of backdrop images issue appears in [Chuang et al. 2000] that uses a color gradient as backdrop pattern. More specifically, a planar slice of the RGB cube was used as the colors in the image. This method is limited to perfectly specular colorless objects. Our method is a straight forward generalization to the single backdrop case where we use unique colors in the RGB cube that are not necessarily limited to a single plane. As a result, we are in a better position to avoid the noisy matte reported in [Chuang et al. 2000].

- We demonstrate the correctness of our approach by comparing our results to images of the models rendered in the same novel environment (assuming known geometry) using a standard rendering software, Persistence-of-View RayTracer. We are able to see desired properties such as colored transparent objects, mirrored surfaces, as well as highlights.

While the environment mattes only provide an approximation to the way an object truly refracts and reflects light, it produces quite convincing and pleasing results. This then is the motivation for our choice of color model.

1.3 Roadmap

The rest of the paper is organized as follows. In Section 2, we present the mathematical preliminaries that form the basis of the solution to the problem. Section 3 introduces the main idea behind our approach. Section 3.1–3.5 details our algorithm for capturing and rendering colored, reflective, and refractive objects in arbitrary environments. The discussion and illustration of our results are provided in Section 4. We conclude with some ideas of future work in Section 5.

2 Problem Formulation

We follow the development in [Chuang et al. 2000]. Consider an object O in a (cubic) environment map B. The (possibly virtual) camera records, for each pixel p a value C from a blend of pixels in B. The set of pixels which contribute to p is called the *footprint* [Wexler et al. 2002] of p. We can describe an environment as light $\mathbf{E}(\omega)$, coming from all directions ω. Thus we have the vector equation,

$$C = \int \mathbf{W}(\omega)\mathbf{E}(\omega)d\omega \tag{1}$$

The weighting function $\mathbf{W}$ comprises all means of light transport of environment lighting from all directions ω through a foreground object to the camera. Note that this equation holds under the assumption that there is no wavelength coupling (e.g. fluorescence).

We now rewrite this equation as a spatial integral over a bounding surface, an environment map $\mathbf{T}(x)$ (in our case, a cube environment map). Further the equation is augmented to include an additive foreground color F. This foreground color models object emissivity, ambient illumination, and any additional reflection from lighting in the scene that is separate from the environment map. Under these assumptions, the equation becomes,

$$C = F + \int \mathbf{W}(x)\mathbf{T}(x)dx \tag{2}$$

From this, various equations can be developed that allow a foreground object to be embedded in a new environment with varying degrees of quality. If no assumptions are made about the nature of the weighting function, then we can write a matrix equation $\mathbf{C} = \mathbf{F} + \mathbf{LB}$ where $\mathbf{L}$ represents a scene dependent transfer matrix that represents all possible light transport. At the other extreme, if only a small axis aligned region in the plane facing the camera (the backdrop) is assumed to impact the foreground image, then we have $C = F + (1-\alpha)B + \Phi$ where α represents the coverage of the pixel (in an input image) by the corresponding point on the object surface and Φ represents the contribution of all refracted and reflected light from the backdrop B through the same pixel. Φ constitutes the positional information (region in the backdrop) of a contributing pixel, and also the amount of contribution.

Chuang et al.'s [2000] formulation generalizes the weighting function for a ***single*** (side of the cube environment) texture map $\mathbf{T}(x)$ as a summation of Gaussians $\mathbf{G}_i(x)$, n representing the number of contributions from a texture map $\mathbf{T}(x)$ and R_i being an attenuation factor,

$$\mathbf{W}(x) = \sum_{i=1}^{n} R_i \mathbf{G}_i(x) \tag{3}$$

Here the summation is over n different portions of a backdrop background image. For modeling the complete environment, as required in Eqn. 2, Chuang et al. [2000] independently model ***three*** sides (a backdrop and two sidedrops) of the cube environment map. Therefore, their weighting function is:

$$\mathbf{W}(x) = \mathbf{W}_1(x) \bigcup \mathbf{W}_2(x) \bigcup \mathbf{W}_3(x) \tag{4}$$

Here we use the union to indicate that the computation happens independently over the different back and side drops. Substituting Eqn. 3 & 4 into Eqn. 2,

$$C = F + \sum_{i_1=1}^{n_1} R_{i_1}\mathbf{G}_{i_1}(x)\mathbf{T}_1(x) \bigcup \sum_{i_2=1}^{n_2} R_{i_2}\mathbf{G}_{i_2}(x)\mathbf{T}_2(x) \bigcup \sum_{i_3=1}^{n_3} R_{i_3}\mathbf{G}_{i_3}(x)\mathbf{T}_3(x) \tag{5}$$

A third way ([Wexler et al. 2002]) of thinking about the weighting function (over a single backdrop) is to abandon the parametric Gaussian, and simply write it as

$$C = F + R\sum_{u,v} W(u,v)B(u,v) \tag{6}$$

which is a pixel based explicit representation of the footprint of p. Here $\sum_{u,v} W(u,v) = 1$.

Unlike all other work, our EM model considers the environment encompassing the object to be a cube environment map as a whole. It is represented as $\mathfrak{T}(x)$, where we have six texture maps simultaneously affecting the foreground object. Since we must deal with multimodal distributions, we adopt Equation 6 as our model resulting in

$$C = F + \sum_{i=1}^{m} \mathbf{W}(x_i)\mathfrak{T}(x_i) \tag{7}$$

There are a few points specific to our implementation that is worth noting.

- The paradigm of working with a cube map (to discover coupling) makes the task harder since we have to search over six times the area for every pixel to find the footprint. An efficient solution is a must.
- The weighting function W also takes into consideration the alpha (α) associated with every pixel. Since the calculation of α was shown in [Chuang et al. 2000], we do not focus on computing α except to figure out the silhouette (as in [Zhu and Yang 2004]) Pixels in the interior of the object have $\alpha = 1$. We experimented with multiple values of α for pixels on the object silhouette. In general, we found $\alpha = 0.4$ to be a conservative estimate of those pixels. The remaining pixels have $\alpha = 0$.

  With higher camera resolutions available, we have to do more work. However, we have the advantage that each pixel is smaller justifying the above empirical values.
- As in other work, we place no constraints on the shape, or the index of refraction (ior) of the transparent object. To enable comparison with a ray tracer, we use the same pinhole synthetic camera model. Thus, multiple rays can join (as in Figure 2) and come to a foreground pixel. However, only one ray penetrates the pixel.
- The quantity F is computed similar to the method in [Chuang et al. 2000] with a slight variation (Section 3.4) for the case of the colored transparent object.

3 Our Approach

Determining the matte reduces to determining the contributing pixels in the background (the footprint), and their respective weights. The key idea in our algorithm is to make each contributing pixel of the cube map unique. A naive way of doing this would be to illuminate one pixel at a time and sweep over the object. To cover this area requires too many images – equal to the total surface area. Sweeping a line segment, rather than a pixel at time, and using a hierarchy of colored progressively finer stripe patterns [Boyer and Kak 1987; Caspi et al. 1998; Zhang et al. 2002] reduces the number of required images to $O(\log_c(k))$.

The input to our algorithm is a set of (foreground) images of the scene, composed of the transparent object placed in *patterned* cube maps (aka background image). These images are used for creating

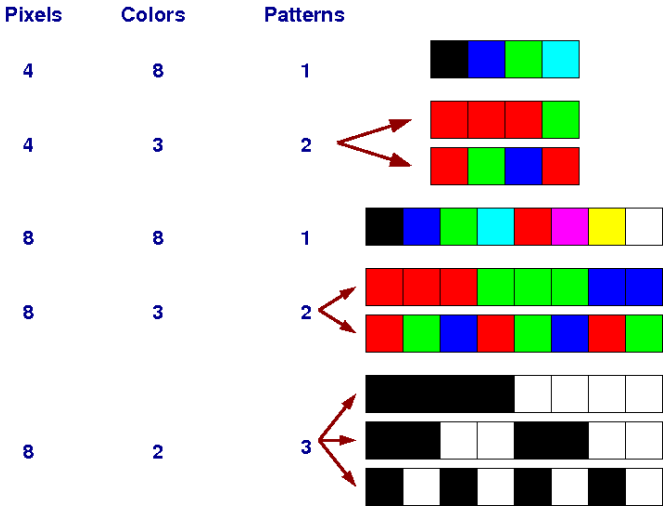


Figure 3: *Number of patterns depends on number of pixels in the background image and number of colors used for coding (Equation 8). In the pattern at the right bottom, the first, second, and third pixels have the codes 000, 001, and 010 respectively.*

a mapping between the cube map pixels and the (input) foreground image pixels. This map is then utilized for relighting and compositing in a novel environment. In the subsequent three sections, we discuss the three primary stages of our algorithm – *generating patterns*, which essentially assigns a unique color code to each pixel of the cube map, *generating the map*, i.e., calculating and storing the pixels from the cube map contributing to each pixel of the foreground image, and *relighting*, i.e., compositing the transparent object in a novel environment.

3.1 Pattern generation

In this section, we detail on the generation of patterns (for the cube map) and their significance. The number of patterns required depends on the number of pixels in the cube map and number of colors (m) used for color coding the pixels. We choose 3-8 colors for generating these patterns. Theoretically, we could assign a unique color to each pixel of the cube map, and thereby have just one pattern in which the transparent object would be captured but that would make the algorithm susceptible to noise. Note the trade off between increasing the number of colors (thereby decreasing the number of patterns and hence the number of input images) and the accuracy of the algorithm. The intuition for the number of patterns required for generating a unique color code for every pixel position in the cube map is given in (Figure 3), and results (for a c-coded decimal instead of a binary coded decimal) in the following number

$$\lceil \frac{\log(|Pixels|)}{\log(|Colors|)} \rceil = \lceil \frac{\log(k \times k \times 6)}{\log c} \rceil \qquad (8)$$

Using the above formula, N patterns are generated which are used as cube maps for enclosing the transparent object, under which images of the scene are captured. The captured N images are our input images, which are used for generation of the mapping (between the cube map pixels and the input image pixels).

For example, the number of patterns required for a cube map (with each face) of resolution 512×512 and 3 colors for coding is 13 (Equation 8). Images of the scene composed of the transparent object and these 13 (cube map) patterns are captured and used for the next stage of the algorithm, finding maps. Note that the next two stages of our algorithm (finding maps and relighting) is independent of the resolution of a face of the cube map, but is dependent on the resolution of the input images.

3.2 Map Generation

Let us consider the case when only one pixel in the enclosing cube map contributes to the color of a pixel in the (input) foreground image. Each pixel of a foreground image is observed across all patterns. The sequence (color code) observed at each pixel is unique, and hence we can find out the corresponding pixel of the cube map. This case (when only one pixel in the background cube map affects the pixel in the foreground image) occurs when the transparent object has purely refractive properties. As only one of the light rays present in the environment, after getting refracted through the object, can reach a particular pixel in the foreground image (ray has to pass through both the pixel and the camera pinhole), color of the pixel is determined by the color of that (single) ray. We term this as the case of *purely refractive or reflective* objects. A diagrammatic illustration of map generation is shown in Figure 4(a).

In the general case of transparent objects with refractive and reflective properties, each pixel in a foreground image is influenced by multiple pixels in the cube map, each contributing in definite (Figure 4(b)) proportions. We take into consideration the contribution of the "major" pixels (ones with proportions above a threshold) in the cube map, and thus, find the map for all the pixels in the foreground image using the following recursive dominance heuristic.

The *recursive dominance principle* states that if pixels $p_i, i \in \{1, 2, \ldots, n\}$, with contribution values $v_i, i \in \{1, 2, \ldots, n\}$ (in decreasing order) respectively, superpose to form a foreground pixel's color, then $v_i > \sum_{j=i+1}^{j=n} v_j, \forall i \in \{1, 2, \ldots, n-1\}$ In the context of transparent objects, this principle can be physically interpreted to mean that the contribution of a refracted ray is much greater than the contribution of a reflected ray, which in turn is greater than the contribution of the reflected ray bounced twice, and so on. Our experiments and results are also shown to support this heuristic (Section 4). When only two pixels contribute to the foreground, the principle holds trivially. Using this, and based on the principle of superposition, we have the pseudocode for the (general case of) map generation.

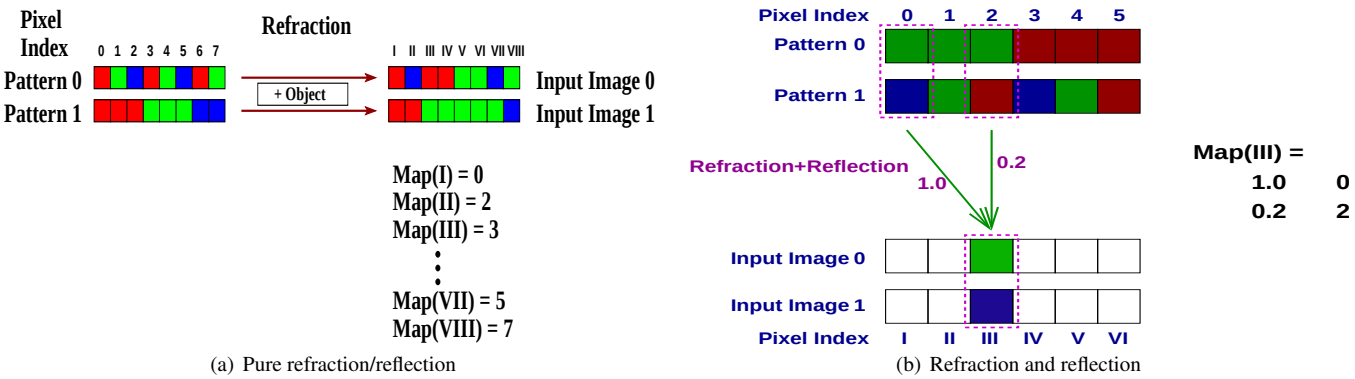
Algorithm 3.2: **Map-Generation** (`foreground images`)

1. Use three-colored (red, green, blue) patterns for generating input images.
2. For each pixel in the foreground image,
   - Find the major contributing pixel in the background image.
   - Store the index of this pixel and its contribution.
   - Subtract the above contribution from the observed values.
   - Repeat the above steps until there doesn't exist any major contributing pixel.

The map calculated using Algorithm 3.2 is then used for compositing the transparent object in novel environments. Finding an exact match in Step 2 for the major contributing pixel is of course not expected – a pixel using the least square measure can be found, however.

3.3 Relighting

Relighting and compositing takes an environment cube map as input and uses the *map* generated to relight the object. For each pixel of the new image to be composited, we look up the map for pixels



(a) Pure refraction/reflection (b) Refraction and reflection

Figure 4: *Finding the map, detecting the pixels in the background image (cube environment) which affected the (input) foreground image pixel. In case of purely refractive/reflective objects, a pixel in the foreground image is affected by only one pixel of the background image. In the general case (refractive and reflective objects), multiple pixels of the background image contribute to a foreground image pixel.*

in the cube map (and their corresponding proportions) which contribute to the color. The final color of each pixel in the composite is determined by combining colors of the contributing pixels from the novel cube map in their respective proportions.

So, the algorithm of capturing the transparent object and rendering it in a novel environment is as follows:

Algorithm 3.3: The-Algorithm (`foreground images, novel environment`)

1. Given the resolution of (each face) of the cube map and the number of colors to be used for coding, generate appropriate patterns.
2. Using the patterns generated in Step 1, capture images of the transparent object in the patterned cube maps.
3. Apply Algorithm 3.2 to generate the map between each pixel position in the foreground images and the pixels in the cube map.
4. Using the map generated in Step 3 and the specified novel environment, relight and compose the transparent object into the scene.

3.4 Colored Objects

Transparent objects, with spatially varying unknown color, pose a problem because, now the color of the light changes as it passes through the object. Since the colors of a foreground image pixel are filtered and modified, it is not possible to discover the major pixel that contributes to the image. We adapt the solution given in [Chuang et al. 2000].

First, instead of using arbitrary colors for the pattern, we choose monochrome colors. This of course means that we have logarithm to the base two in Equation 8, and thus a larger number (18 instead of 7) of background images. In addition, we also generate three cube maps with all faces having black (assumed intensity 0), white and gray color respectively.

Next, we capture images I_j with the generated patterns and Q with the solid black cube map. Recall that there is an ambient lighting factor. As a result, Q contains pixels with values 0 (if the object is unaffected by the ambient light), or colors β representing the true object colors (we simply do not know the mapping yet). Consider the image obtained by subtracting Q from one of the images I_1. Now the pixels in the resultant image are either black (value 0), or some other non-zero value. If the value of a pixel is 0, we know that this pixel must have come from only the black pixels in the patterned images. The remaining pixels get their contribution from the white pixels in the patterned cube maps. By proceeding through the other patterned images I_2 and so on, we are able to pinpoint the exact pixel of the environment map responsible for the matte.

However, we still have no knowledge of the object's color. This is where the second solid image (with white background) contributes. It gives us an estimate of the true color of the object because multiple contributing pixels could saturate the color. Therefore we use a third (gray) image and compute the color as follows. Let N be the color of a pixel in the *novel* environment map affecting a pixel p in the foreground image. Let b, gc, and w be the foreground images obtained by rendering the object inside a solid black cube map, solid gray cube map, and white cube map respectively. We observe that the final color of p can be computed using:

$$C(p) = b(p) + N * (w(p) - b(p)) + \delta \tag{9}$$

where $\delta = gc(p) - b(p) - 0.5 * (w(p) - b(p))$. The first term tells us the ambient illumination. The second term gives the emissivity of the object, plus light reflected from the environment, and the third term is a correction term for saturation. The *equivalence* of this with prior methods when white is 1 and black is 0 is shown in the Appendix; however, our formulation is slightly more robust to saturation effects.

Figure 6: *Rendered images of a transparent wine glass, with refractive and reflective properties. Not the highlights created due to the presence of bright point light sources in the scene (best viewed in color).*

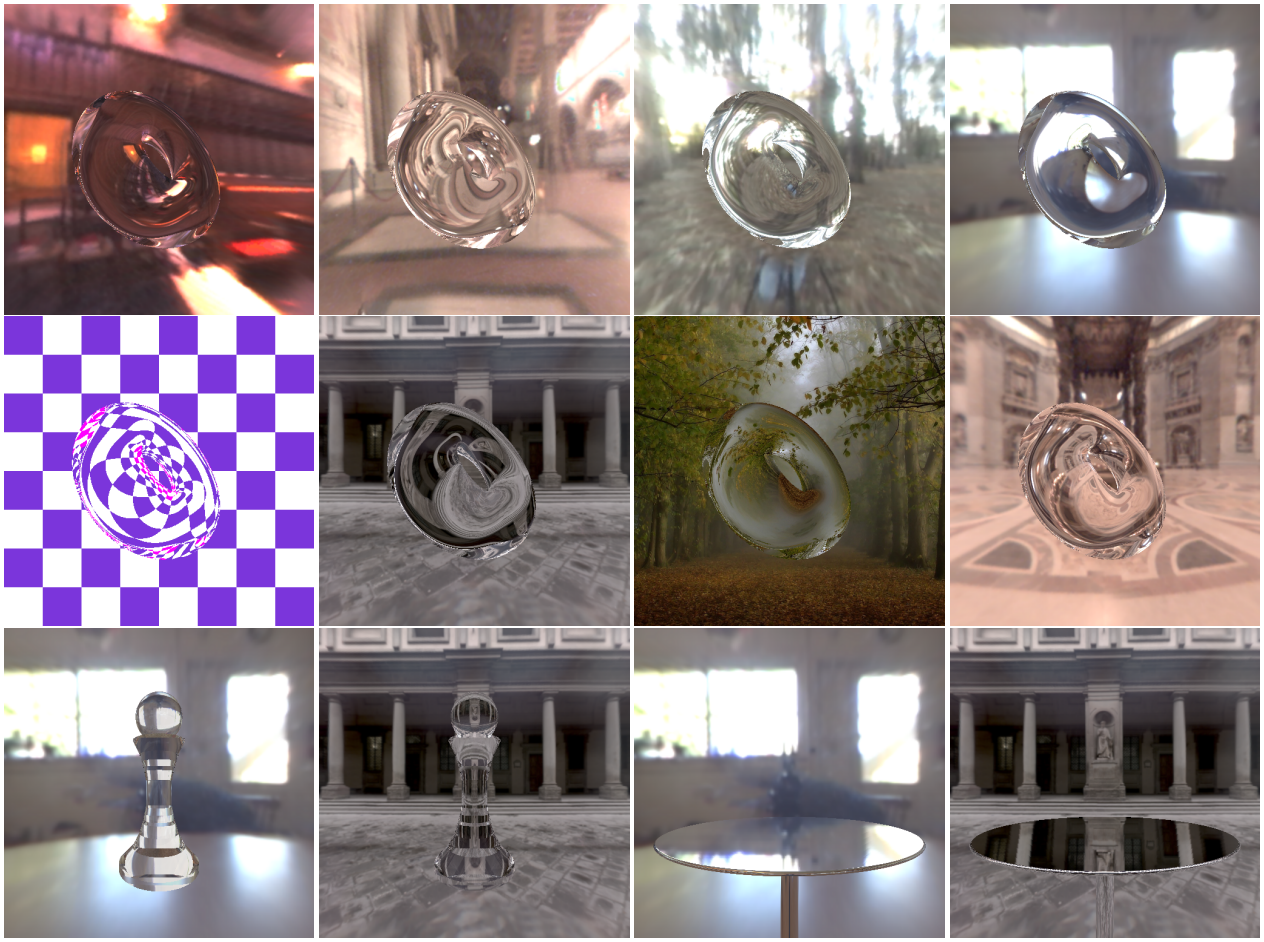


Figure 5: *Rendered images of a transparent torus and a chess pawn, exhibiting only refractive properties; and a purely reflective table, digitally composit in novel environments (best viewed in color).*

3.5 Highlights

One of the most noticeable visual effect of transparent objects are specular highlights. The objects we typically capture have curved surfaces, and so highlights are important both, for better understanding of the object shape and its visual appeal. In this section, we develop a method for recovering and generating highlights for objects exposed to additional lighting in the scene apart from the cube environment map.

Unlike the colored case, highlights are modeled as the foreground color F of the object (with the restriction that it is white) and therefore is additive to the computed map (as in Eqn 7). For any object exhibiting highlights, we capture a single image H of the object exposed to bright, point-light sources and a solid black cube map enclosing it. From Eqn. 7, we see that the observed color $C = F$. Since we use point light sources, the only lit part in H are the highlights. Subtracting H from each of our input foreground image, would give us a set of images having only the influence of the cube environment map. Applying Algorithm 3.2, we compute the map and thereby relit and composit the object into a novel environment. Now, to create highlights on the object, we add the image H. Results for this case are in the additional material (it's easier to appreciate this in an electronic form).

4 Implementation and Results

For our novel environments, we use the data provided in [Debevec], namely *The Uffizi Gallery, Florence*, *St. Peter's Basilica, Rome*, *Grace Cathedral, San Francisco*, *Eucalyptus Grove, UC Berkeley*, *Kitchen at 2213 Vine St*, *Galileo's Tomb, Santa Croce, Florence*, and also generated two other environments (chess board, and woods).

For purpose of comparison with ground truth, we generated results for a *purely refractive torus* (with ior 1.5), which was relit and composited in novel environments (Figure 5). The images obtained are **exactly** the same as obtained using POV-Ray. The backdrop of the chessboard is made up of violet and white, whereas the sidedrop contains pink color; these are faithfully reproduced in the torus. Thus even in the purely refractive case, it is possible to have colors from the sidedrops.

Next, we show the results (Figure 7) obtained for the general case of multiple refraction, and reflection. We exhibit the torus, a horse, and a wine glass. We also demonstrate results using a colored torus and the dragon captured and relit under different environments (Figure 8). For results, please check the additional material.

We also have produced results to demonstrate real time acquisition of objects. For cases when we cannot afford to stop the motion of an object, we use a single image (each pixel composed of an

Figure 7: *Rendered images of transparent torus, horse & glass, exhibiting reflective and refractive properties, under novel environments (best viewed in color).*

unique color) as a background image for data capture. As a proof of concept, images of a purely refractive dice moving at 25fps along an arbitrary path against a colored pattern were analyzed. Based on this, the dice is composit in novel environments. For results, please check the additional material.

All computations and timing calculations have been done with MATLAB on a Dual-Core AMD processor with 2GB RAM. Typically, we require around 30 seconds to compute the matte for purely refractive, reflective and colored objects of size 512×512. (An optimized C version is expected to take substantially less time.) The time needed for compositing in the same categories is around 4–7 seconds. For objects exhibiting both refractive and reflective properties, the pre-processing and compositing time was found to be approximately 70 and 20-30 seconds respectively. Time taken by our algorithm is independent of the geometry complexity of the object (since we do not use any geometric information), but is dependent on the (background and foreground) images' resolution. This was verified by performing experiments with two other image resolutions, 256×256 and 1024×1024.

5 Conclusion and Future Work

Environment matting and compositing techniques involves an inherent tradeoff between the amount of input data required and the quality of the discovered matte. Most of the previous impressive techniques have come at the cost of unduly large amount of input data. For a detailed perceptual discussion of accuracy in transparent objects, please refer [Khan et al. 2006].

In this paper, we present a novel environment matting and compositing technique which uses *color structured environment maps* as cues and performs adequately with a very small number of images. The approach allows us to capture objects exhibiting multimodal refractive and reflective properties. It also allows for selective attenuation of light (for colored transparent objects) and highlights. We also present a single-image method of computing mattes, which enables us to capture the behavior of moving refractive elements.

Like most other techniques, our work assumes a fixed viewpoint. In the future, it would be useful to have high quality real time uncalibrated high dynamic range input data that can be processed in real time and enable varying view points.

Acknowledgements

References

BATLLE, J., MOUADDIB, E. M., AND SALVI, J. 1998. Recent progress in coded structured light as a technique to solve the

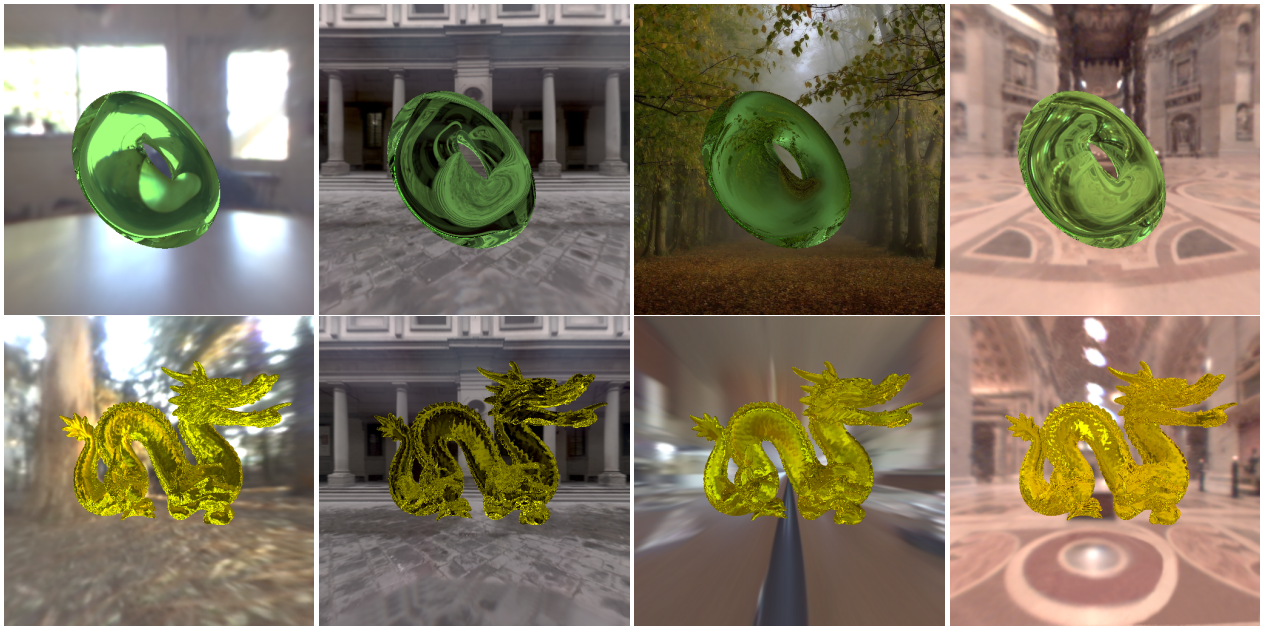


Figure 8: *Rendered images of a colored transparent torus & dragon under novel environments (best viewed in color).*

correspondence problem: a survey. *Pattern Recognition 31*, 7, 963–982.

BESL, P. J. 1988. Active, optical range imaging sensors. *Machine Vision and Applications 1*, 2, 127–152.

BLINN, J. F., AND NEWELL, M. E. 1976. Texture and reflection in computer generated images. *Communications of the ACM 19*, 10, 542–547.

BOYER, K. L., AND KAK, A. C. 1987. Color-encoded structured light for rapid active ranging. *IEEE Transactions on PAMI 9*, 1, 14–28.

CABRAL, B., OLANO, M., AND NEMEC, P. 1999. Reflection space image based rendering. In *SIGGRAPH '99*, 165–170.

CASPI, D., KIRYATI, N., AND SHAMIR, J. 1998. Range imaging with adaptive color structured light. *IEEE Transactions on PAMI 20*, 5, 470–480.

CHOUDHURY, B., AND CHANDRAN, S. 2006. A survey of image-based relighting techniques. In *GRAPP '06*, 176–183.

CHUANG, Y.-Y., ZONGKER, D. E., HINDORFF, J., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2000. Environment matting extensions: towards higher accuracy and real-time capture. In *SIGGRAPH '00*, 121–130.

CHUANG, Y.-Y. 2004. *New models and methods for matting and compositing*. PhD thesis, University of Washington.

DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., AND KOENDERINK, J. J. 1999. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics 18*, 1, 1–34.

DEBEVEC, P. Light probe image gallery. `http://www.debevec.org/Probes/`. Last visited: June, 2007.

DEBEVEC, P., HAWKINS, T., TCHOU, C., DUIKER, H.-P., SAROKIN, W., AND SAGAR, M. 2000. Acquiring the reflectance field of a human face. In *SIGGRAPH '00*, 145–156.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *SIGGRAPH '96*, 43–54.

HEIDRICH, W., AND SEIDEL, H.-P. 1999. Realistic, hardware-accelerated shading and lighting. In *SIGGRAPH '99*, 171–178.

HEIDRICH, W., LENSCH, H., COHEN, M. F., AND SEIDEL, H.-P. 1999. Light field techniques for reflections and refractions. In *EGRW '99*, 187–196.

KAY, D. S., AND GREENBERG, D. 1979. Transparency for computer synthesized images. In *SIGGRAPH '79*, 158–164.

KHAN, E. A., REINHARD, E., FLEMING, R. W., AND BÜLTHOFF, H. H. 2006. Image-based material editing. *ACM Trans. Graph. 25*, 3, 654–663.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *SIGGRAPH '96*, 31–42.

MARSCHNER, S. R. 1998. *Inverse rendering for computer graphics*. PhD thesis, Cornell University. Adviser-Donald P. Greenberg.

MATUSIK, W., PFISTER, H., ZIEGLER, R., NGAN, A., AND MCMILLAN, L. 2002. Acquisition and rendering of transparent and refractive objects. In *EGRW '02*, 267–278.

PEERS, P., AND DUTRE, P. 2003. Wavelet environment matting. In *EGRW '03*, 157–166.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. In *SIGGRAPH '84*, 253–259.

SATO, K., AND INOKUCHI, S. 1985. Three-dimensional surface measurement by space encoding range imaging. *Journal of Robotic Systems 2*, 1, 27–39.

SMITH, A. R., AND BLINN, J. F. 1996. Blue screen matting. In *SIGGRAPH '96*, 259–268.

VOORHIES, D., AND FORAN, J. 1994. Reflection vector shading hardware. In *SIGGRAPH '94*, 163–166.

WEXLER, Y., FITZGIBBON, A. W., AND ZISSERMAN, A. 2002. Image-based environment matting. In *SIGGRAPH '02*, 198–198.

ZHANG, L., CURLESS, B., AND SEITZ, S. M. 2002. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *IEEE International Symposium on 3D Data Processing, Visualization, and Transmission*, 24–36.

ZHU, J., AND YANG, Y.-H. 2004. Frequency-based environment matting. In *PG '04*, 402–410.

ZONGKER, D. E., WERNER, D. M., CURLESS, B., AND SALESIN, D. H. 1999. Environment matting and compositing. In *SIGGRAPH '99*, 205–214.

APPENDIX A

Chuang et al. [2000] solves for the foreground color $F(p)$ of a pixel p by exposing with two solid backgrounds, say B_1 and B_2 using the formula,

$$C(p) = F(p) + R(p) * B \tag{A-1}$$

So the two equations for the two unknowns are

$$C_1(p) = F(p) + R(p) * B_1$$
$$C_2(p) = F(p) + R(p) * B_2$$

where $C_1(p)$ and $C_2(p)$ are the observed color of the pixel under the two solid backdrops. Given a novel backdrop N to composit the object, the authors use the same formula with the computed foreground color F. We have

$$C_{new}(p) = F(p) + R(p) * N \tag{A-2}$$

Our formulation for colored objects (in Section 3.4) is:

$$C(p) = b(p) + N * (w(p) - b(p)) + \delta \tag{A-3}$$

where, $w(p)$ and $b(p)$ are images of the object captured under a white and black cube map respectively. We show that our formulation results in the same result as that of Chuang et al. [2000] under standard assumptions. Using Eqn. A-1,

$$w(p) = F(p) + R(p) * W \tag{A-4}$$
$$b(p) = F(p) + R(p) * B \tag{A-5}$$
$$\delta = (F(p) + R(p) * G) - (b(p) + G * (w(p) - b(p))) \tag{A-6}$$

where W,B and G are environments (cube-maps, in our case) with solid colors white, black and grey repsectively. Substituting the values of Eqn. A-4, A-5 & A-6 into Eqn. A-3,

$$\begin{aligned} C(p) &= (F(p) + R(p) * B) + N * R(p) * (W - B) \\ &\quad +((F(p) + R(p) * G) - (b(p) + G * (w(p) - b(p)))) \\ &= (F(p) + R(p) * B) + N * R(p) * (W - B) + F(p) \\ &\quad +R(p) * G - (F(p) + R(p) * B + G * R(p) * (W - B)) \\ &= F(p) + N * R(p) * W - N * R(p) * B + R(p) * G \\ &\quad -G * R(p) * W + G * R(p) * B \end{aligned} \tag{A-7}$$

In the standard assumption $W = 1$ and $B = 0$, Eqn. A-7 becomes,

$$C(p) = F(p) + N * R(p) \tag{A-8}$$

We observe that Eqn. A-2 and A-8 are identical.