Automatic Spatially Varying Illumination Recovery of Indoor Scenes Based on a Single RGB-D Image

Guanyu Xing, Yanli Liu, Haibin Ling, Xavier Granier, and Yanci Zhang

Abstract—We propose an automatic framework to recover the illumination of indoor scenes based on a single RGB-D image. Unlike previous works, our method can recover spatially varying illumination without using any lighting capturing devices or HDR information. The recovered illumination can produce realistic rendering results. To model the geometry of the visible and invisible parts of scenes corresponding to the input RGB-D image, we assume that all objects shown in the image are located in a box with six faces and build a planar-based geometry model based on the input depth map. We then present a confidence-scoring based strategy to separate the light sources from the highlight areas. The positions of light sources both in and out of the camera's view are calculated based on the classification result and the recovered geometry model. Finally, an iterative procedure is proposed to calculate the colors of light sources and the materials in the scene. In addition, a data-driven method is used to set constraints on the light source intensities. Using the estimated light sources and geometry model, environment maps at different points in the scene are generated that can model the spatial variance of illumination. The experimental results demonstrate the validity and flexibility of our approach.

Index Terms—Illumination recovery, Automatic, Indoor scenes, Single RGB-D image.

1 Introduction

PHOTO-realistic integration between virtual objects and images or videos is required in applications such as visualization and augmented reality, where recovering the illumination distribution from real scenes plays an important role. Although cameras with high dynamic range (HDR) sensors can capture an HDR image from a single or several exposures [1], [2], in practice, for convenience, most people capture low dynamic range (LDR) images when recording real scenes. Therefore, illumination recovery based on a single LDR image is highly desirable in practice.

Numerous methods have been proposed to acquire the illumination of real scenes [3], and impressive results have been achieved. However, most of these methods use either extra equipments, such as light probes [4], [5] or expensive customized cameras [6], require user input [7], [8], or assume that a series of images of the scene were available [9], [10]. These conditions limit the applicability of these methods. RGB-D images are getting easier to acquire even with hand-held devices. Consumers products, such as Kinect, provide this kind of information. Furthermore, more and more smart phones embed two cameras that may be used for stereo reconstruction. In this paper, we address the problem of recovering lighting based on a single LDR RGB-D indoor scene image, without requiring any additional equipment or assistance from users.

As we know, the observed color of each point in a scene is the result of complex interactions between scene geometry, material, and the illumination condition. The information contained in a single LDR RGB-D image is inherently incomplete for lighting estimation, which causes the problem ill-posed. The first challenge lies in the fact that a single RGB-D image cannot record portions of a scene out of the camera's view, which makes 3D geometry reconstruction and further lighting recovery difficult. Secondly, even when light sources are captured, both the light source pixels and the highlight pixels are usually saturated (because the light sources are too bright to be captured by an LDR image), making it difficult to distinguish between light sources and highlights from a single image. Moreover, lacking knowledge of the material in a scene causes the determination of the light source intensities an ill-posed problem, even when the scene geometry and the positions of light sources are known. Last but not least, indoor illumination often varies greatly from one location to another in real scenes. While this property has been widely utilized in the application of indoor lighting design, recovering spatially varying illumination is still a challenging problem.

This paper proposes an automatic framework to recover the illumination distribution from an RGB-D image of an indoor scene. To obtain the scene geometry behind the camera, we coarsely synthesize a 3D model of the invisible scene based on the input RGB-D image, under the assumption that all objects shown in the image are located in a box with six faces (a common assumption for indoor scenes).

- *Guanyu Xing is with National Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University, Chengdu, China. He is also with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China. E-mail: xgyuestc@gmail.com.*
- *Yanli Liu is with College of Computer Science, Sichuan University, Chengdu, China. E-mail: yanliliu@scu.edu.cn.*
- *Haibin Ling is with Department of Computer and Information Sciences, Center for Data Analytics and Biomedical Informatics, Temple University, Philadelphia, PA, USA. E-mail: hbling@temple.edu.*
- *Xavier Granier is with the LP2N laboratory, Institut d'Optique Graduate School, Bordeaux-France. E-mail: xavier.granier@institutoptique.fr*
- *Yanci Zhang is with College of Computer Science, Sichuan University, Chengdu, China. E-mail: yczhang@scu.edu.cn.*

Manuscript received November 15 2017.

Then, we propose a confidence-scoring based strategy to separate saturated pixels into light sources and highlight areas, which helps in detecting light sources both in and out of the camera's view. Using the recovered model, an iterative procedure is developed to estimate the intensity of light sources and scene materials by adopting spherical harmonics, in which a data-driven method is used to set constraints on the light source intensities. To further improve the efficiency of illumination estimation, an acceleration strategy is introduced. Finally, using the recovered light sources and the geometry model, we generate 2D environment maps at different points in the scene and use them to render virtual objects at different positions.

The main contributions of this paper are as follows. (1) We propose a confidence-scoring based strategy to separate pixels of a saturated region into light sources and highlights by exploiting both color and geometry features. (2) We introduce a reliable approach to detect light sources both in and out of the camera's view. (3) We present an iterative strategy to estimate the intensity of light sources, in which a data-driven method is used to set constraints on the light source intensities. These contributions lead to a novel framework that automatically estimates the spatially varying illumination of indoor scenes from a single RGB-D image. Experimental results have validated the effectiveness of the proposed method.

2 Related Works

Current methods for recovering illumination from images can be roughly classified into two categories: those that recover the illumination condition at a specific point and those that recover the illumination of the entire scene. The former category is inconvenient for embedding virtual objects into a scene, particularly at different positions. The latter category can solve the above problems; however, the existing methods require special equipments, multiple images or HDR information to recover the illumination.

Assuming that the scene geometric model is known, some researchers tried to directly estimate the illumination utilizing shadow cues [11], critical points [12], etc. Mei et al. [13] found that images produced by a Lambertian scene can be efficiently represented by a sparse set of images generated by directional light sources. To obtain the illumination from images with unknown geometry, Lopes-Moreno et al. [5] presented a method to estimate multiple light sources from a single image. One object in the image is selected as a virtual light probe; then, the direction and intensity of the light sources are calculated based on the silhouette of the light probe. Unfortunately, all the methods mentioned above ignore spatially varying illumination. Reinhard [14] mapped the background image area around an inserted virtual object to a hemisphere to form environments both in front of and behind the image plane. However, this work does not recover any HDR information, which causes the light source intensities to be inaccurate. More recently, Karsch et al. [15] proposed an automatic strategy for estimating illumination using a single image of a Lambertian scene that adopts a data-driven approach to estimate the out-of-view light sources. However, this method ignores occlusions of visible light sources, and the invisible light sources and environmental light are recorded by only one environment map. Thus, it still needs improvement to be able to handle spatially varying illumination. Moreover, the invisible scene is recovered according to the best matched luminaire-annotated panorama of the background image selected from a panorama data set. It cannot be demonstrated theoretically that the date-driven strategy of recovering invisible scene mentioned in their work can detect accurately light sources which are not appeared in the image.

Unger et al. [16], [17], [18] conducted a series of works on dense spatial light sampling to record the spatially varying illumination in 1D, 2D and 3D. They use HDR video cameras, light probes and customized special equipments in all their methods. To capture illumination based on sparse sampling, Sato et al. [19] employed two omni-directional cameras to generate the spatial radiance distribution of the environment using stereo matching; the resulting 3D mesh with lighting information can provide spatially varying illumination for rendering scenes. Banterle et al. [20] introduced EnvyDepth that allows users to "paint" depth onto an HDR environment map to create a rough scene model that captures both geometry and illumination. Meilland et al. [21] proposed a system based on dense real-time 3D tracking and mapping with an RGB-D camera. The dense scene structure is estimated simultaneously with the observed dynamic range to compute a radiance map of the scene and fuse a stream of low dynamic range images (LDR) into an HDR image.

However, all the methods discussed above to recover spatially varying illumination require special devices or multiple images of the examined scene to be captured; thus, they are unsuitable for the problem of estimating lighting based on a single LDR image. Hara et al. [22] attempted to estimate the surface reflectance property of an object as well as the position of a light source from a single view without an assumption of distant illumination. Unfortunately, this method requires that the scene contains only a single point light source, which limits its application. The authors of [7], [8], [23] explored how to estimate spatially varying lighting conditions based on a legacy LDR photograph. However, [8] is designed for outdoor scenes and models only the sun and sky as light sources. Moreover, these three methods all require users to provide the geometry and light sources of the scene.

3 Indoor Illumination Model

Objects in indoor scenes receive direct illumination from light sources and indirect illumination from the environment, both of which can be regarded as area light sources distributed over a sphere. Therefore, we can record the light at a point p using an environment map and denote its distribution as D_p. Our approach adopts the assumption that scene materials are diffuse, which is also used in other illumination estimation methods [5], [7], [15]; thus, we ignore specular reflection. Adopting the rendering equation proposed in [24], the appearance of p in image I is:

$$I_p(\lambda) = \rho_p(\lambda) \int_{\Omega(\mathbf{n}_p)} D_p(\omega, \lambda)(\mathbf{n}_p \cdot \omega) d\omega \qquad (1)$$

where λ denotes the R, G, B channels, ρ_p is the diffuse coefficient, $\mathbf{n}_p$ is the surface normal at p, $\Omega(\mathbf{n}_p)$ is the upper

hemisphere over the surface at where p locates, and ω is a unit direction vector.

We adopt spherical harmonics Y_{lm}, with $l \geq 0$ and $-l \leq m \leq l$, to approximate the integral in Eq. 1. It has been shown that I_p can be represented by a linear combination of Y_{lm} [24]

$$I_p(\lambda) = \rho_p(\lambda) \sum_{l,m} \hat{A}_l L_{lm,p}(\lambda) Y_{lm}(\mathbf{n}_p) \tag{2}$$

where $\hat{A}_l$ is related to the combination coefficients of $A = (\mathbf{n}_p \cdot \omega)$, which is a constant, and $L_{lm,p}(\lambda)$ can be calculated as follows [8]:

$$L_{lm,p}(\lambda) = \sum_{(\theta,\phi)\in\Omega_p} D_p(\lambda,\theta,\phi) Y_{lm}(\theta,\phi) \sin\theta \Delta\theta\Delta\phi \tag{3}$$

Here, Ω_p is the unit sphere surface, which can be divided into $\Omega_p^{<s,i>}, i \in \{1,2,\cdots,n\}$ and Ω_p^{env} corresponding to the n light sources and surrounding objects, respectively. To analyze the light sources and indirect illumination separately, we write $L_{lm,p}(\lambda)$ as:

$$\begin{aligned} L_{lm,p}(\lambda) &= \sum_{i=1}^{n} L_{lm,p}^{<s,i>} + L_{lm,p}^{env} \\ &= \sum_{i=1}^{n} \sum_{(\theta,\phi)\in\Omega_p^{<s,i>}} D_p(\lambda,\theta,\phi) Y_{lm}(\theta,\phi) \sin\theta\Delta\theta\Delta\phi \\ &+ \sum_{(\theta,\phi)\in\Omega_p^{env}} D_p(\lambda,\theta,\phi) Y_{lm}(\theta,\phi) \sin\theta\Delta\theta\Delta\phi \end{aligned} \tag{4}$$

We assume the color of each light source is homogeneous. Our experiments show that this approximation can reasonably keep the accuracy of the estimation results, while at the same time save a lot of computations. Denote L_i^s as the color of the ith light source, and set $R_{lm,p}^{<s,i>} = \sum_{(\theta,\phi)\in\Omega_p^{<s,i>}} Y_{lm}(\theta,\phi)\sin\theta\Delta\theta\Delta\phi$ from Eq. 4. Then, we have:

$$L_{lm,p}(\lambda) = \sum_{i=1}^{n} L_i^s(\lambda) R_{lm,P}^{<s,i>} + L_{lm,p}^{env}(\lambda) \tag{5}$$

Substituting this back into Eq. 2, we obtain

$$\begin{aligned} I_p(\lambda) =& \rho_p(\lambda)\Big(\sum_{i=1}^{n} L_i^s(\lambda) \sum_{l,m} \hat{A}_l R_{lm,P}^{<s,i>} Y_{lm}(\mathbf{n}_p) \\ &+ \sum_{l,m} \hat{A}_l L_{lm,p}^{env}(\lambda) Y_{lm}(\mathbf{n}_p)\Big) \end{aligned} \tag{6}$$

Finally, by calculating $P_p^{<s,i>} = \sum_{l,m} \hat{A}_l R_{lm,p}^{<s,i>} Y_{lm}(\mathbf{n}_p)$ and $E_p^{env}(\lambda) = \sum_{l,m} \hat{A}_l L_{lm,p}^{env}(\lambda) Y_{lm}(\mathbf{n}_p)$, our indoor illumination model at point p is

$$I_p(\lambda) = \rho_p(\lambda) \sum_{i=1}^{n} P_p^{<s,i>} L_i^s(\lambda) + \rho_p(\lambda) E_p^{env}(\lambda) \tag{7}$$

4 Indoor Lighting Recovery

This section describes our lighting recovery method. We separate the input scene into visible and invisible parts respectively corresponding to the areas in and out of the camera's view. We first propose a new approach for synthesizing a geometry model containing both visible and invisible parts of the scene based on the input depth map. Then, we discuss how to estimate the positions of light sources. Finally, an iterative procedure is presented to calculate the colors of the light sources. Based on the recovered geometry model and light sources, an environment map can be created for any point in the scene.

4.1 Geometry Modeling

Similar to [7], we assume that all the objects visible in the input image are located in a closed hexahedral box; therefore, they can be represented by the composition of a few planes, which is a reasonable assumption for indoor scenes. Compared to triangle-based models, this planar model requires considerably less computation to determine the cross points, which makes generating an environment map more efficient.

For the visible parts of the scene, the depth map provides sufficient information to reconstruct the 3D geometry. Specifically, we first cluster pixels based on their normal directions and 3D positions using mean shift. The pixels belonging to the ith cluster are used to calculate a plane P_i using the method proposed in [25]. All the recovered planes are added to a plane set S_p. Fig. 1 (b) and Fig. 2 (b) show the recovered planar models of two test scenes.

It is impossible to recover an exact model of the invisible scene, because of the large ambiguity. To address this issue, we assume that no extra objects exist behind the camera. Combined with the closed hexahedral box assumption, the problem of invisible scene reconstruction is converted to that of recovering the box in which the scene is located. The six faces of the box will correspond to the scene's floor, ceiling, and walls. We can adopt RGB-D semantic segmentation methods [27], [28], [29] to find such faces, however, the implementations of these methods are usually difficult. In this work, we propose a simple but effective approach that selects several planes from S_p to assemble the box based on the fact that all the scene objects are located on the same side of each face. If a box face is not found in S_p, we set it as a directly designed plane. Scene textures are also generated based on the input image. The details of our geometry modeling method can be found in Sec. 5. Fig. 1 (c) and Fig. 2 (c) show the visible part of the recovered box. The invisible part of the box can be found in the recovered environment maps, which are demonstrated in Fig. 1 and Fig. 2.

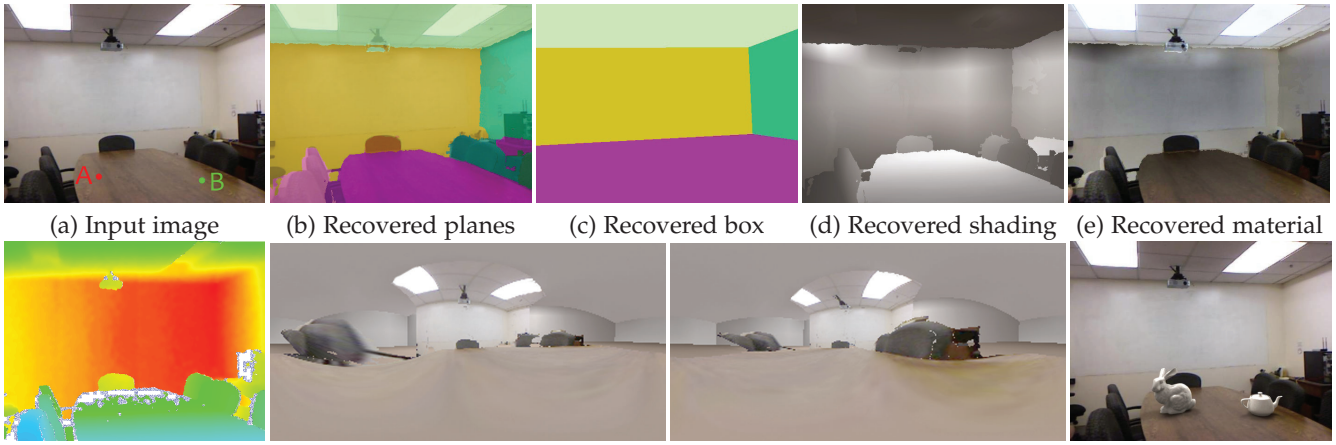
4.2 Estimating Light Source Positions

Light sources in an indoor scene are usually too bright to be captured by a camera, which suggests that we could assume that saturated pixels are likely to be light sources. However, highlights will also cause saturated pixels. Therefore, the saturated pixels in an image can be divided into two classes that correspond to light sources and highlights respectively. In the remainder of this section, we first discuss how to distinguish whether a pixel belongs to a light source or a highlight area; then, we introduce our method for calculating the light source positions.

(a) Input image (b) Recovered planes (c) Recovered box (d) Recovered shading (e) Recovered material

(f) Input depth (g) Environment map at point A (h) Environment map at point B (i) Synthetic result

Fig. 1. A test scene from the SUN RGB-D dataset [26] captured by a Kinect V2: (a) the input RGB image; (b) the recovered planar model of the visible scene; (c) the visible part of the recovered box; (d) the shading image calculated by the recovered illumination and 3D scene model; (e) the material of the scene estimated according to the shading image; (f) the input depth data; the white areas correspond to pixels whose depths are not captured; (g) and (h) the recovered environment maps at point A and B, respectively. The two points are demonstrated in (a); (i) the final synthetic result, the swivel chair and white torus are the two virtual objects rendered by the environment maps shown in (g) and (h), respectively.



(a) Input image (b) Recovered planes (c) Recovered box (d) Recovered shading (e) Recovered material

(f) Input depth (g) Environment map at point A (h) Environment map at point B (i) Synthetic result

Fig. 2. A test scene from the NYU dataset [27] captured by a Kinect V1: (a) the input RGB image; (b) the recovered planar model of the visible scene; (c) the visible portion of the recovered box; (d) a shading image calculated by the recovered illumination and the 3D scene model; (e) scene material; (f) input depth data; (g) and (h) recovered environment maps at points A and B, respectively; and (i) the final synthetic result.

4.2.1 Pixel Classification

Didyk et al. [30] have proposed a method to classify a saturated region into light source and highlight. However, in practice light source and highlight may be mixed in a same saturated area. Such as the first image of Fig. 3, highlights located around the lamp and the lamp together form a saturated area in the image. Didyk's method cannot handle this situation. In our work, a confidence-scoring based strategy, which adopts both color and geometry features of a region, is presented to solve the problem.

We consider pixels whose maximum color channel is greater than 250 (the color channel value range is $[0, 255]$) and whose minimum color channel is larger than 230 as saturated pixels. Pixels belonging to an upward plane are discarded, because light sources are rarely embedded into surfaces such as floors or table tops. Shen's method [31] is adopted to detect the highlight pixels. Together highlight pixels and saturated pixels form a pixel set D that includes all the light sources and highlights in an image.

We first cluster the pixels in D. Pixels located in a connected area are composed as one class. We then give each class a score based on the fact that light sources usually have indistinct boundaries in an image, while the boundaries for highlight areas are fuzzier. The score of the ith connected area C_i is calculated as follows:

$$\sum_{p \in B(C_i)} w(p) Var_{bright}(N(p))$$

where $B(C_i)$ is the set of C_i's boundary points, $N(p)$ is p's neighborhood, and we set its radius to $0.005 * (image_{width} + image_{height})$; $Var_{bright}(S)$ indicates the brightness variance of pixel set S (the pixel values range is $[0, 1]$ when calculating the variance), and there will be a small value for a pixel set located in a highlight area boundary. $w(p)$ is a weight function defined as follows:

$$w(p) = \frac{w_1(p) \cdot w_2(p)}{\Sigma_{q \in B(C_i)} w_1(q) \cdot w_2(q)}$$

TABLE 1
The pixel classification strategy in C_i, where t_{high} and t_{low} are two thresholds.

score	classification strategy
$S(C_i) > t_{high}$	All pixels are labeled as light source
$S(C_i) < t_{low}$	All pixels are labeled as highlight
$S(C_i) \in [t_{low}, t_{high}]$	Containing light source and highlight pixels

We call $w_1(p)$ a geometry constraint item; its value is calculated by

$$w_1(p) = \frac{\sum_{q \in N(p)} |\mathbf{V}_p \cdot \mathbf{V}_q|}{\|N(p)\|}$$

where $\mathbf{V}_p$ is the normal vector of p, and $\|N(p)\|$ is the size of $N(p)$. This item mainly penalizes the pixels located at the edge of the geometry, because the variations around such pixels may not be due solely to changing illumination.

We call $w_2(p)$ a material constraint item. Because the differences in pixel values caused by material are usually larger than those caused by illumination, we select only pixels whose neighborhoods have a smaller variance value to estimate the final score. Here, $w_2(p)$ is calculated as follows:

$$w_2(p) = \begin{cases} 0 & Var_{bright}(N(p)) > \tau \\ \tau - Var_{bright}(N(p)) & Var_{bright}(N(p)) \leq \tau \end{cases}$$

where τ is a threshold. In this study, we set τ to $1.5\frac{\sum_{p \in B(C_i)} Var_{bright}(N(p))}{\|B(C_i)\|}$.

There are three possibilities for the ith connected area C_i: (1) all pixels in C_i belong to a light source; (2) C_i contains both light source pixels and highlight pixels; and (3) all pixels in C_i are highlight pixels. We denote the score of C_i as $S(C_i)$. Unsaturated pixels in C_i will be labeled as highlight, while the remaining pixels are labeled as listed in Table 1. When $S(C_i) \in [t_{low}, t_{high}]$, the pixels near C_i's center point will be labeled as light source pixels. To avoid the size of a light source too small, the proportion of light source pixels in C_i is set by $max(0.1, \frac{S(C_i)-t_{low}}{t_{high}-t_{low}})$. Through our experiments, the upper bound t_{high} is set to 0.1, and the lower bound t_{low} is defined as $min(0.06, max(0.026, 0.4 * aver_s))$, where $aver_s$ is the average score of all the connected areas. The second column of Fig. 3 demonstrates the results of our pixel classification algorithm.

4.2.2 Calculating Light Source Positions

Light sources may be located both in and out of the camera's view; called visible and invisible light sources, respectively. We introduce a new way to estimate the positions of these two types of light sources in the rest of this section.

Visible Light Sources. Visible light sources exist in the input image; therefore, all pixels previously labeled as light sources are classified as visible light sources. To simplify the algorithm, light source pixels belonging to the same connected area form one light source, and the position of that light source is calculated using the recovered geometry model.

Invisible Light Sources. Light from light sources reflected by a specular surface will cause a highlight when the camera is looking in the direction of the reflected ray. We can roughly estimate the position of invisible light sources by the highlights shown in the image. However, pixels labeled as highlight may be caused by either visible light sources or erroneous detection, which will make the detection result inaccurate. To remove such pixels, we first discard the pixels belonging to areas that contain both light source pixels and highlight pixels, because highlights often appear around light sources. Considering a pixel p from the remaining highlight pixels, we calculate the reflection ray r_p of the gaze direction at p. If r_p intersects with a visible part of the scene or floor, p will be abandoned. If r_p intersects with the invisible scene, but the intersection point is too distant from p, p will also be abandoned. This restriction is mainly due to the limited range of indoor light sources. The distance is set to 6 meters in our work. Points without depth values are discarded as well, because these points cannot supply exact geometry information. The remaining highlight points that form a point set S_h are used to estimate the positions of the invisible light sources. Suppose there are n connected areas belonged to S_h, and denote the ith connected highlight area as S_h^i which corresponds to a single light source. The third column of Fig. 3 shows the pixels adopted to estimate invisible light sources. Our method can reject most errors from detected highlight pixels while preserving highlight areas caused by invisible light sources.

For a highlight point $\tilde{p} \in S_h^i$ caused by the ith invisible light source, we calculate the reflection ray $\tilde{r}_p$ of the gaze direction $\tilde{g}_p$ at $\tilde{p}$, and assume that all invisible light sources are embedded in the walls or ceiling; therefore, the intersection point $\hat{p}$ of $\tilde{r}_p$ and the wall or ceiling belongs to

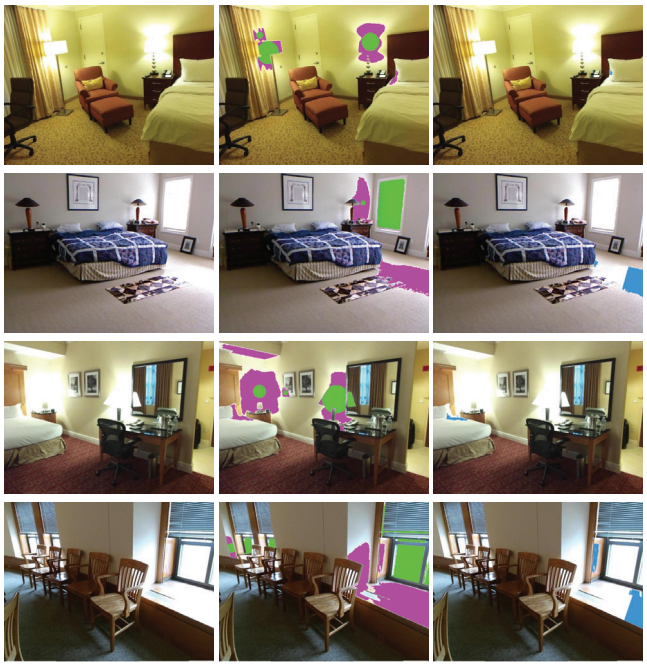


Fig. 3. Pixel classification results and the highlight areas adopted to estimate invisible light sources. The first column shows the input images; the second column indicates our pixel classification results (purple for highlight areas and green for light sources). The third column demonstrates the highlight areas adopted to estimate the invisible light sources (blue pixels).

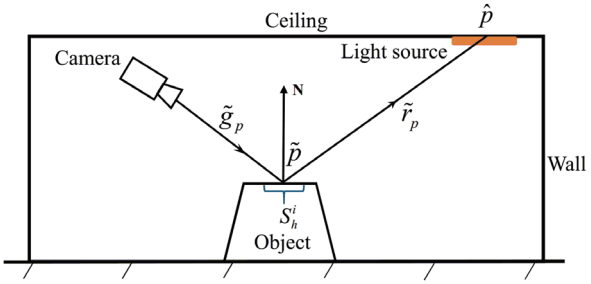


Fig. 4. Geometric configuration to detect the light source position from its resulting highlight in the RGB-D image.

the ith invisible light source (Fig. 4). Obviously, all points belonging to S_h^i together decide the shape and position of the ith light source. To ensure that the invisible light source forms a connected area, wall or ceiling points around the recovered position are also set as part of the light source.

Fig. 5 illustrates the positions of light sources recovered by the described method. We adopt two virtual scenes here, therefore, the ground truths of all light sources are available. The comparisons between the recovered light sources and the ground truths show that our method can capture most of the light sources in the scene, and the positions of light sources in the recovered environment maps are very close to those in the ground truths.

4.3 Estimating Light Source Colors

Looking back to Eq. 7, for point p in the visible scene, $P_p^{s,i}$ can be calculated according to the position of the ith light source. E_p^{env} can also be estimated based on the scene model. From this point on, the unknowns in Eq. 7 are the material parameters and light color parameters. An iterative procedure is proposed to calculate materials in the scene and the colors of light sources. The values of these unknown parameters are solved based on several sampled pixels. We first remove the highlight and light source pixels because they do not satisfy our illumination model. Then, we sample pixels uniformly. The set of all sampled pixels is denoted as N_s in the rest of this section.

4.3.1 Material Parameter Estimation

If we know the colors of all the light sources, an objective function based on the theory of intrinsic images [32] can be developed to solve the material ρ of all the points in N_s .

$$argmin_\rho \sum_{\lambda\in\{R,G,B\}} \sum_{p\in N_s} (\rho_p(\lambda)K_p(\lambda) - I_p(\lambda))^2 + \mu \sum_{\lambda\in\{R,G,B\}} \sum_{p,q\in N_s} w(p,q)\cdot(\rho_p(\lambda) - \rho_q(\lambda))^2 \tag{8}$$

where $K_p(\lambda) = \sum_{i=1}^n P_p^{<s,i>} L_i^s(\lambda) + E_p^{env}(\lambda)$; μ is a weight parameter; it is set to 400 in our experiments; $w(p,q)$ is a weighting function defined as follows:

$$w(p,q) = \exp(-0.02\cdot\|I_p - I_q\|^2)$$

The first term in Eq. 8 is the data term, which ensures that the synthesized image conforms with the original image; the second term is the global smoothness term, which penalizes pixel pairs with different RGB values.

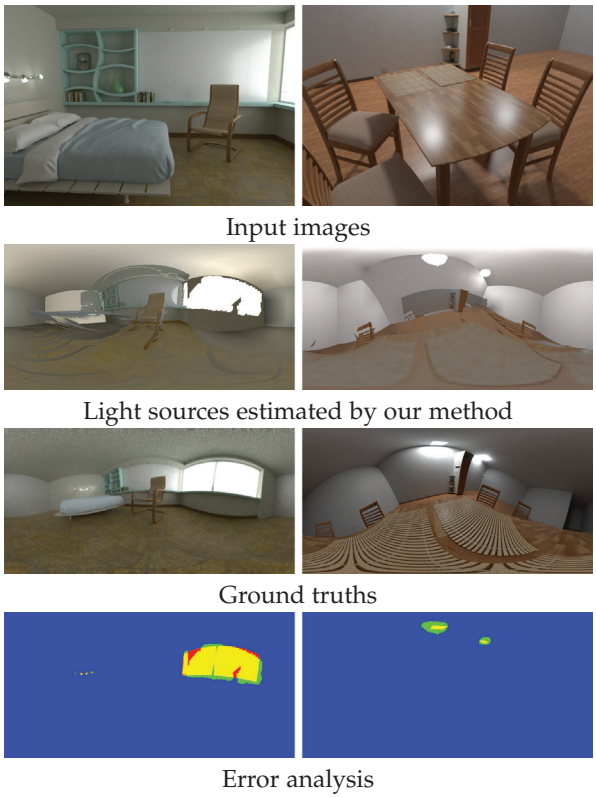


Fig. 5. The accuracy of our light source position calculation method. The first row demonstrates the input images; the second and third rows compare our recovered light sources with the ground truths; the last row illustrates the results of accuracy evaluations of the recovered light sources positions: the red and green pixels correspond to the light source areas belonging to the ground truth and to our results respectively, while the yellow pixels indicate the light source area overlap of the ground truth and our method.

4.3.2 Calculating Light Source Colors

Using the recovered material, the light color of each light source can be calculated by solving a linear system of equations. Additional constraints are adopted to provide a lower bound for the light color parameters.

We first investigate the luminance of the light sources and the objects in real scenes. A data driven method is proposed to estimate their luminance. We use 26 HDR environment maps of real scenes. Two thresholds are set to divide each environment map into areas corresponding to objects, highlights and light sources. Based on the fact that light sources and highlights are much brighter than other part of a scene, the first threshold τ_1, which is used to separate objects and highlight areas, can be estimated by as follows:

$$\tau_1 = min(2\cdot l_{ave}, \frac{(l_{min} + l_{max})}{2}) \tag{9}$$

where l_{ave} is the average luminance of the environment map, and l_{min} and l_{max} are the minimum and maximum luminance value of the environment map.

Similarly, light sources are much brighter than highlights. Therefore, the second threshold, τ_2, is the average luminance of the pixels whose values are larger than τ_1; pixels whose luminance are greater than τ_2 will be considered as light sources. The detection results are shown in

Environment maps Light sources and highlights

Fig. 6. The detected light sources and highlights of two HDR images. The white areas in the second column indicate the positions of light sources and the grey pixels correspond to highlights.

Without constrains With constrains

Fig. 7. Virtual objects rendered under illuminations calculated with and without using the constraint of light color.

Fig. 6, where the white pixels indicate the positions of light sources, and the grey pixels correspond to highlights.

From the HDR data, the average luminance ratio between the objects and light sources of our environment map data set is 76.6. The brightness of the pixels in LDR image corresponding to objects with normal light exposure is approximately 122.5; therefore, the average brightness of light sources for an LDR image should be $\eta = 76.6 \times 122.5 \approx 9384$. We set η as the lower bound of all invisible light sources.

For a visible light source s_{vis}, its lower bounders for different channels should relate with its color in the image. Thus, the value of its channel λ, $\lambda \in \{R, G, B\}$ should be larger than $L_{low}(s_{vis}, \lambda) = f_{vis}(s_{vis}) \cdot Ave_p(s_{vis}, \lambda)$, in which $Ave_p(s_{vis}, \lambda)$ is the average value of all the pixels belonging to s_{vis} in the λ channel, and $f_{vis}(s_{vis})$ is defined as

$$f_{vis}(s_{vis}) = \max(1, \min(\kappa, (\kappa - 1) \cdot \frac{Ave_p^{min}(s_{vis}) - 230}{250 - 230} + 1)) \quad (10)$$

where $Ave_p^{min}(s_{vis}) = \min(Ave_p(s_{vis}, \lambda))$, $\lambda \in \{R, G, B\}$; κ is the ratio factor. We set $\kappa = 38.3$, due to the reason that the values of saturated pixels are usually twice 122.5. To penalize light source whose pixels' values are smaller than 250, $f_{vis}(s_{vis})$ is set to a value smaller than κ. The lower bound of all the light sources are used to set the initial light parameters of the entire iterative procedure. Fig. 7 illustrates virtual objects rendered under illuminations calculated with and without using the constraints of light colors. The virtual statue looks inconsistent with the background image if we do not adopt the color constraints.

4.3.3 *Updating Model Parameters*

After each iteration, we obtain new illumination parameters; thus, the material of the visible scene can be updated by Eq. 7. The material of the invisible scene is then calculated using a texture generation strategy similar to that introduced in Sec. 5.3. Fig. 1 (e) and Fig. 2 (e) show the recovered material of the visible part of the scene. We can also render the recovered scene model using the estimated illumination. Fig. 1 (d) and Fig. 2 (d) demonstrate the rendered shading image based on the geometry model. The scene update causes a change in the value of E_p^{env}; therefore, we must recalculate E_p^{env} for each $p \in N_s$.

Unfortunately, there are usually over $10,000$ points in N_s; consequently, calculating E_p^{env} for every point is time-consuming. To solve this problem, we recover the environment maps and update E_p^{env} at only several sampled points in the scene. The E_p^{env} of points in N_s can be calculated by bilinear interpolation based on each point's four nearby sampling points.

The method for selecting sampling points is as follows. We find the intersection point of the scene geometry model and the ray using $(0, 0, 0)$ as the origin and $(0, 0, 1)$ as the direction. The z value of the intersection point is denoted as $p_i(z)$. The central point p_c of the scene is set to $(0, 0, 3)$ when $p_i(z) > 6$; otherwise, p_c is set to $(0, 0, 0.5p_i(z))$. Sample directions as all combinations of polar angles from $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and azimuth angles from $[0, 2\pi]$, at intervals of $\frac{\pi}{6}$; shooting rays from p_c along all sampling directions. The intersection points between all rays and the geometry model constitute the set of sampling points.

4.4 Recovering Spatially Varying Illumination

To model spatially varying illumination, one can model light sources as area sources, since their shapes, positions and intensities have already calculated by the illumination modeling method. Combining with the estimated geometry model, virtual objects can be rendered through global illumination rendering technology. As the global illumination rendering is generally time-consuming and the image based lighting methods save a lot of rendering time, we choose an image based lighting method to illuminate the virtual objects in this work. We simply need to generate environment maps according to the recovered planar model and light sources at appropriate points in the scene. The final environment maps created by our method are demonstrated in Fig. 1 (g, h), Fig. 2 (g, h) and Fig. 8. More results of our lighting recovery algorithm can be found in Sec. 6.

5 Geometry Modeling Details

This section provides the details of our scene reconstruction method. We first introduce the camera calibration strategy adopted in our work and then discuss the implementation details of the scene geometry model creation algorithm. Finally, scene texture generation is described at the end of this section.

5.1 Camera Calibration

To reconstruct planar scene models, the first step is to calibrate the camera. The RGB-D images used in this paper

Fig. 8. Recovered environment maps of four test scenes.

were captured by a Kinect; therefore, we set the focal length of the camera to 519 (the Kinect's focal length). For images captured in other ways, the focal length can be calculated by computing a simple pinhole camera as proposed by [33]. We also let the world and the camera share the same coordinate system, therefore the extrinsic parameters matrix is an identity matrix. The 3D position and normal vector of each pixel can then be calculated easily using the camera parameters and the recorded depth map.

5.2 Geometry Modeling

An input depth map provides a reliable geometry model of the visible scene; therefore, the major geometry modeling challenge involves creating the geometry for the invisible scene. As mentioned in Sec. 4.1, the reconstruction of invisible scene is equivalent to that of recovering the box where the scene was located. Our method considers only an image captured by a forward-facing camera (the most common way people take a photo in real life). The rectangular coordinate system in this paper is set so that the positive direction of the x-axis is to the right of the camera, the positive direction of the y-axis is down, and the positive direction of the z-axis points toward the front. We next describe how to estimate the floor, ceiling and walls.

Floor Estimation. The floor must face up; therefore, we select the planes whose normal vectors have y components smaller than -0.8 from S_P as the candidates. To select the correct plane, we assume that all the objects in the scene are located above the floor; thus, the selected plane cannot occlude any object in the scene. However, errors in the recovered planar model may not meet this assumption. This paper introduces a new strategy to solve the problem. We give each selected plane P_k a score calculated as follows:

$$Score(P_k) = \sum_{P_i \in S_P, P_i \neq P_k} \frac{Occlu(P_i)}{Num(P_i)} \cdot \min(1, \frac{Num(P_i)}{0.1 \cdot imgSize}) \quad (11)$$

where $Num(P_i)$ indicates the number of pixels P_i contains, $imgSize$ is the size of the input image, and $Occlu(P_i)$ represents the number of pixels belonging to P_i but occluded by P_k. We do not consider pixels whose depth values are not recorded when calculating $Occlu(P_i)$. From Eq. 11, the combination of the number of pixels in P_i and the ratio of occluded pixels in P_i determine P_i's contribution to P_k's score: P_i has less contribution when it contains few pixels, and the larger the part of P_i that is occluded by P_k, the larger the contribution provided by P_i is. A plane that occludes only a small object or small parts of several objects has a higher probability of being the floor plane; this formula gives such plane a low score. Therefore, we select planes whose scores are smaller than 0.3 from among all the candidate planes. If no plane is chosen, we set the floor plane to $y = max(1.6, 1.5 \cdot maxY)$ (with normal as $(0, -1, 0)$), where $maxY$ is the maximum y value of all points in the visible scene. When only one plane is selected, we set that plane as the floor directly; when more than one plane remains, we adopt the one that contains the most pixels.

TABLE 2
When a wall cannot be found in S_P^{wall}, we can set it as a plane based on this table. Here, $minX$ is the minimum x value of all points in the visible scene, while $maxX$ and $maxZ$ are the maximum x and z values respectively.

wall	corresponding plane
left wall	$x = 1.5 \cdot minX$ with $(1, 0, 0)$ as normal
front wall	$z = 1.5 \cdot maxZ$ with $(0, 0, -1)$ as normal
right wall	$x = 1.5 \cdot maxX$ with $(-1, 0, 0)$ as normal
back wall	$z = -2$ with $(0, 0, 1)$ as normal

Ceiling Estimation. The procedure for estimating the ceiling is quite similar to that of estimating the floor. The differences are that we need to select planes whose normal vectors have y components larger than 0.8. When no plane is chosen according to the scores, we set the ceiling plane to $y = \min(-1.5, 1.5 \cdot minY)$ (with normal as $(0, 1, 0)$), where $minY$ is the minimum y value of all points in the visible scene.

Wall Estimation. There are four walls in the scene: front, left, right, and back walls. The absolute value of the normal vector's y component of a wall should be small. We select planes whose normal vectors have y components larger than -0.5 and smaller than 0.5 as candidates, and implement the same operations used for floor estimation. The acquired plane set of this procedure is denoted as S_P^{wall}. When two planes in S_P^{wall} share similar normals, the one containing fewer pixels is removed from S_P^{wall}. While S_P^{wall} contains more than three planes, we select the two planes whose normal vectors differ the most. Then, for each unselected plane, we estimate the angles between it and the two selected planes and calculate their sum; the plane with the largest summed value is adopted as the third wall.

We keep only these three selected planes in S_P^{wall}. From this point forward, up to three planes remain in S_P^{wall}, each of which may be the left wall, front wall or right wall. We can register them according to their normal vectors. Walls that are not in S_P^{wall} are considered directly as planes, as shown in Table 2.

5.3 Texture Generation

For the visible portion of the scene, we simply map the image to the model based on the transformation from camera space to world space.

For textures in the invisible scene, we assume that each face of the box is composed of uniform material. For an indoor scene, at least one wall should appear in a photo captured by a forward-facing camera. We first find the wall

Fig. 9. Virtual objects are inserted into images selected from SUN RGB-D data set [26]. The original picture is inset above the result.

with the largest visible area in the input image and then train a Gaussian model of that wall. The color of the out-of-view part of the selected wall is set as C_{mean}, which is the mean of the Gaussian model. A wall will not supply reliable color information when only a small part of that wall appears in the image. The appearance of the remaining three walls is calculated by

$$C_{wall} = w_{wall} \cdot C_{wall}^{init} + (1 - w_{wall}) \cdot C_{mean} \qquad (12)$$

where C_{wall}^{init} is the color of current wall estimated by the method for obtaining C_{mean}, and w_{wall} is the weight parameter of the current wall, defined as follows:

$$w_{wall} = \begin{cases} \frac{Num_{wall}^{vis}}{0.1 \cdot imgSize} & Num_{wall}^{vis} < 0.1 \cdot imgSize \\ 1.0 & Num_{wall}^{vis} \geq 0.1 \cdot imgSize \end{cases}$$

where Num_{wall}^{vis} is the number of pixels belonging to the current wall, and $imgSize$ is the size of the image. The appearance of a wall with only a small part appears in the image will be decided jointly by that wall and C_{mean}.

This method is also used to recover the ceiling and floor colors. The only difference is that C_{mean} changes to $0.5C_{mean}$ when calculating the floor's color because floors are usually darker than walls in practice. The appearance of the invisible scene is updated when materials or light source colors are renewed.

6 Experiments

We selected several images from SUN RGB-D data set [26] (captured by a Kinect V2) to test our method. We first recover the illumination of each test scene using our method; then, we generate environment maps at the points where virtual objects will be located. Finally, the virtual objects rendered using our recovered environment maps are embedded into the test image following Debevec's strategy [4]. Windows in a scene are regarded as area light sources in our method. The experimental results demonstrate the validity of this approximation. Fig. 9 demonstrates the synthetic results. As can be seen by the captured spatially varying illumination, the recovered light produces convincing composite results.

Fig. 10. Virtual objects inserted into images selected from the NYU data set [27]. The original picture is inset above the result.

Fig. 11. Embedded animations of a virtual robot into two background images.

We also used images selected from NYU data set [27] to evaluate our method. The NYU data set images were captured by a Kinect V1; thus, more noise is present in the depth map. Fig. 10 shows the final synthetic results, from which we can see that our approach still performs well.

Two animation sequences of a virtual robot were produced to explore how well our method captures spatially varying illumination during movement. Fig. 11 shows some

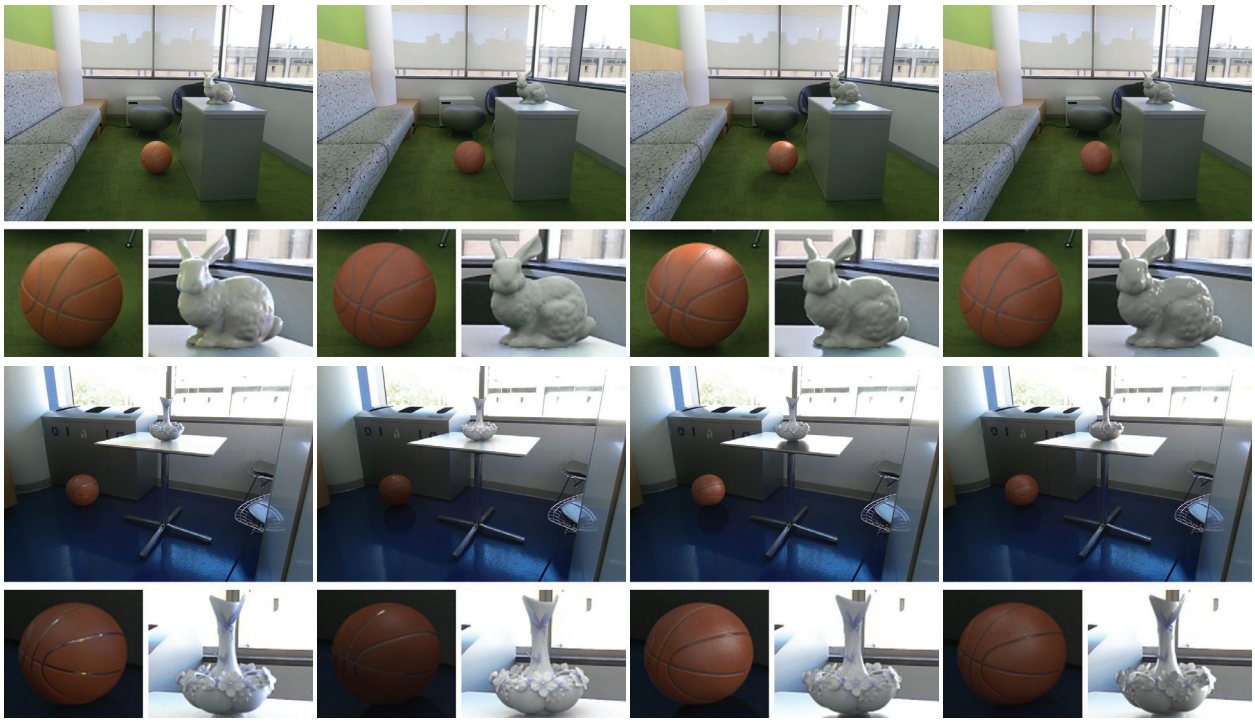


Fig. 12. Comparison between our synthetic results and those of Reinhard's method [14] and Karsch's method [7] and those using a light probe [4]. We insert a bunny and a basketball into the first scene and a basketball and a vase into the second scene. Closeups of the virtual objects are presented under the synthetic results of each scene.

video frames from the synthetic video sequences.

We compared the virtual objects under the illumination recovered by our method with that use a light probe [4], Karsch's method [7] and Reinhard's method [14]. The results are shown in Fig. 12. Reinhard's method maps the LDR background image to a hemisphere without calculating the intensities of saturated pixels, which causes the rendered virtual objects to lose highlight areas. Karsch's method requires user to provide geometry and light sources of the scene corresponding to the input image, which produces more realistic rendering results. However, the positions of highlights on the virtual objects are not very accurate comparing to the results generated by the light probe. Our method estimates the intensity of light sources automatically and produces HDR environment maps for different positions in the scene. Although the results still lose some detail, the virtual object shading is close to that of the results generated by the light probe.

To further compare our method and Karsch's method [7], we adopted two rendered images whose light sources are partly visible or totally invisible. We recover the illumination conditions of these two images by our method and Karsch's method, and render virtual objects (the white bunny and two balls) using the estimated illuminations. Ground truths can be also created by rending the virtual scenes. The comparison results are demonstrated in Fig. 13, from which we can see that the appearances of virtual objects under the illumination recovered by Karsch's method are different from the ground truths. This is mainly because that a user can hardly find the accurate positions of light sources which are not appeared in the image. Our method utilizes

Fig. 13. Further comparison between our synthetic results and those of Karsch's method [7]. We insert a bunny and two balls with different materials into these two images. Closeups of the virtual objects are presented under the synthetic results of each scene.

highlights caused by light sources out of cameras view when estimating the positions of invisible light sources, which is supported by the law of reflection, thus the estimated light sources are more reliable. Therefore, the synthetic results generated by using our method are very close to the ground truths.

We adopted two images taken at different scene posi-

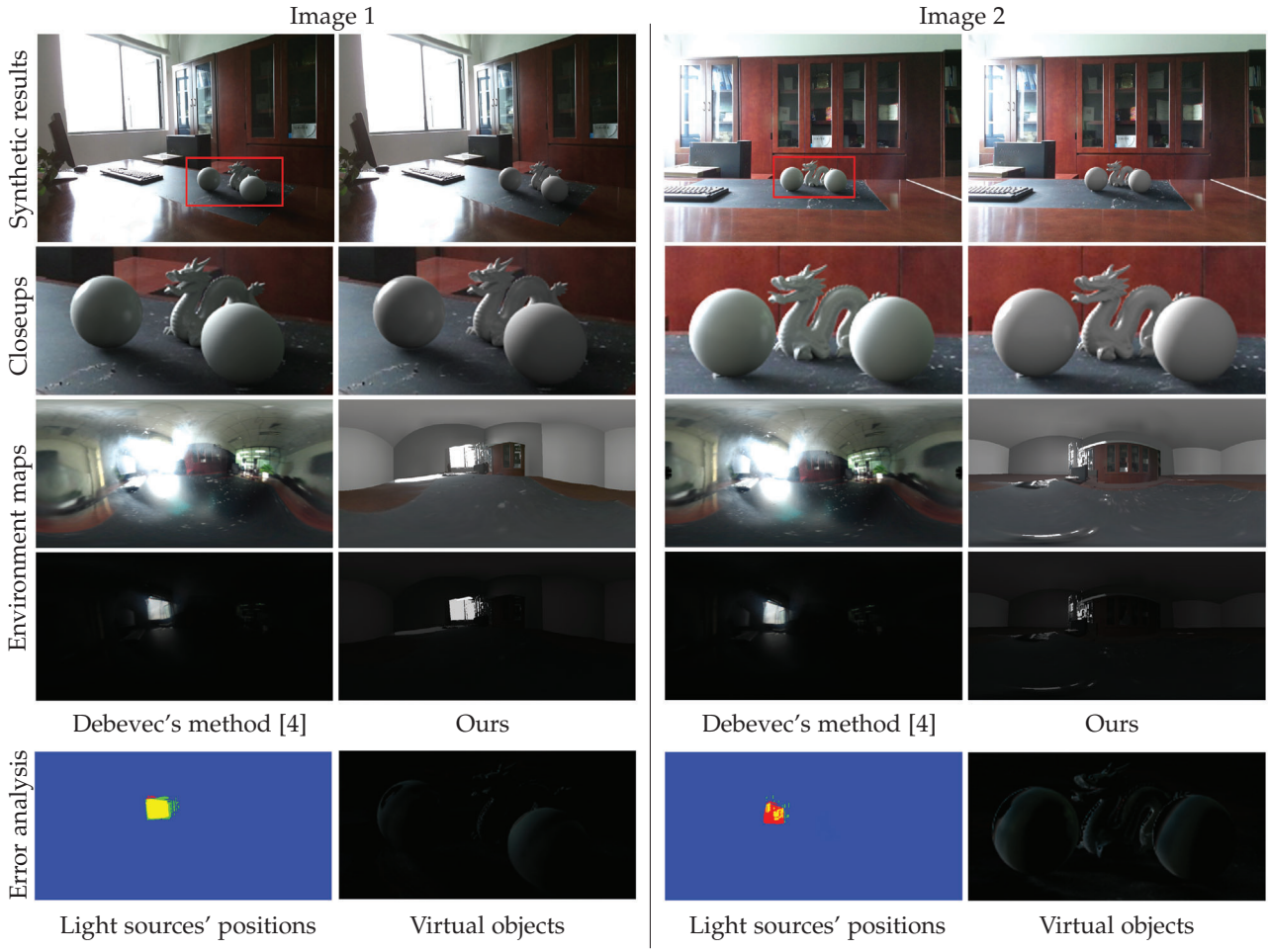


Fig. 14. The accuracy of our light recovery algorithm when the light sources are inside (image 1) and outside (image 2) of the camera's view. The first row demonstrates the synthetic results produced by the ground truth and estimated illumination; the second row shows the closeups of the virtual objects; the third and fourth rows compare the environment maps recovered by Debevec's [4] method and our method at different exposure times. The left figures in the last row illustrate the results of accuracy evaluations of the recovered light sources' positions: the red and green pixels correspond to the light source areas belonging to the ground truth and to our results respectively, while the yellow pixels indicate the light source area overlap of the ground truth and our method. The error maps corresponding to the red boxed areas in the first row are shown in the right figures in the last row.

tions to test the accuracy of our lighting recovery algorithm when the light sources are inside (image 1) and outside (image 2) the camera's view. We followed Debevec's method [4] to capture the illumination conditions at specified positions. This method is based on a light probe that attempts to record the exact illumination of the scene: those closest to the real lighting conditions. The environment maps generated by these two methods should be adjusted to a uniform scale by letting them have similar luminance values. The test results are shown in Fig. 14. For each image, we first render two virtual balls with different materials and a virtual dragon under illumination created by the light probe and our method. The rendered results are then inserted into the background images. The synthetic results are shown in the first row of Fig. 14. From these images, we can see that regardless of whether the light sources exist inside or outside the image, the shadings of our rendered objects are close to the results rendered by the captured environment maps, which indicates that our method produces a correct illumination distribution. We also evaluated the accuracy of our rendered results quantitatively. The right figure in the last row of Fig. 14 illustrates the absolute error of our rendered virtual objects, showing that large errors are mainly located in the highlighted areas of the virtual objects. The relative errors of the RGB channels of the virtual objects are listed in Table 3. Environment maps with different exposures are shown in the third and fourth rows, which reveal the positions and sizes of light sources. For each image, the left figure in the last row shows the errors of the light sources' positions. Our recovered light source areas are closely consistent with the ground truth in image 1. The errors of light sources' shapes occur mainly because a small part of the window is not captured in the background image. More over, the differences between the estimated environment map and the captured one shown in the third row of Fig. 14 are mainly caused by highlights located around the window, however, it impacts a little on the appearances of rendering results due to the fact that light sources are usually much brighter than highlights. For image 2, although our method fails to find all the light source areas, it still detects the main light

TABLE 3
Relative errors of virtual objects rendered under our recovered illumination in Fig. 14.

image	R channel	G channel	B channel
image 1	2.9%	4.7%	4.0%
image 2	4.4%	7.1%	5.8%

source whose luminance in our estimated environment map is brighter than that in the captured one; consequently, it still performs well.

To further examine the accuracy of our illumination recovery results, we compare more synthetic results with real scenes. We used two real white plastic balls as the reference objects, and captured RGB-D images of several indoor scenes with these balls as ground truth images, while the captured images without balls served as background images. Then, we recovered the 3D models of the two balls and inserted them as virtual objects into the background images. We initially assumed that the plaster models were composed of a uniform Lambert material with weak specular reflectance. Then, we adjusted their reflectance coefficients interactively until they matched one of the chosen test images successfully. Finally, the rendered synthetic scenes were compared with the ground truths. We tested 15 different scenes. Fig. 15 shows two scenes of our comparison results. For the ground truth and the synthetic result of the same scene, we also measured the error by comparing the pixel intensity differences on each ball (the intensity value of each pixel has a range of $[0, 1]$). The results are demonstrated in Fig. 15. The average RMSE of all models is 0.099 and the RMSEs of most balls are less than 0.135. However, the RMSE may be affected by errors in the estimated material. Consequently, we measured the RMSE adjusted error [7] to remove the influence of the material to be able to evaluate how accurately the shading distribution on each model is reproduced. We first calculated the mean intensity of each ball and then removed the mean intensity value from each pixel, from both the ground truth image and the synthetic image. Finally, we obtained the RMSE based on the adjusted pixels' values. The experimental data showed a mean adjusted RMSE of 0.087, and the errors of most models are below 0.12.

Our recovered illumination may differ from the real lighting conditions; however, the difference does not preclude obtaining a realistic synthetic result. We conducted a user study to measure how well a user can differentiate between the ground truth and the synthetic result. As subjects, 71 graduate students were recruited for this task; most had a computer science background. Our test used 20 scenes, among which 14 included both a real photo and a synthetic result, while the other 6 contained only real photos or only synthetic results. Fig. 16 shows two test scenes from this user study; the dragon, the red ball and the green ball in the synthetic results are virtual objects. All these scenes were presented in a random sequence. The image placement (left or right) was also randomized. The users were told that there were three possible conditions for each pair of images: (1) one is real photo and the other is synthetic; (2) both are real photos; (3) both are synthetic results. The subjects were given three options: the left image is a real photo, the right image is a real photo, and not sure. The result of this user study is shown in Fig. 16. We found that nearly half the subjects were unable to identify the ground truth in most scenes. Some subjects thought that the synthetic results were more realistic, and most subjects admitted that it was difficult to discern abnormalities in the synthetic images without comparing them to the ground truth.

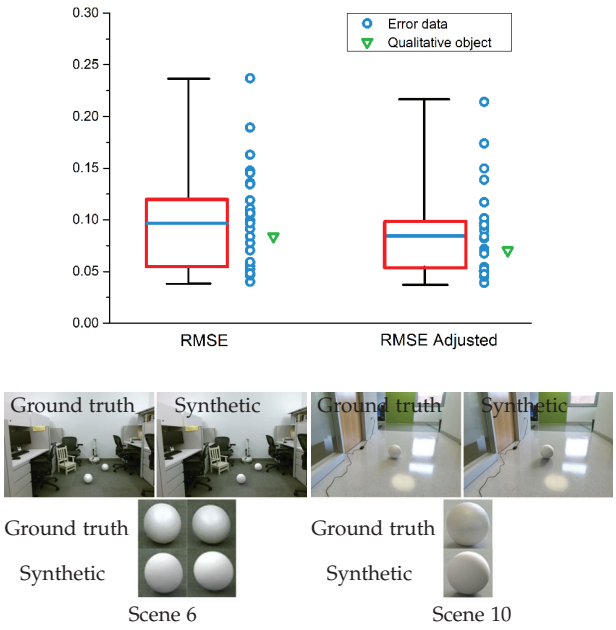


Fig. 15. The root mean squared error (RMSE) and the adjusted RMSE per sphere. For each metric, the blue horizontal line denotes the average error, and the red box indicates the 25% to 75% variance. The error data is represented by the blue circles. The images below the graph show two of the scenes used in this error test. The green triangle indicates the error of the right sphere illustrated in scene 6.

TABLE 4
Time consumptions for solving different number of light sources' colors.

Number of light sources	2	3	6	7	12
Time consumption (seconds)	112	133	165	203	308

We implemented the proposed approach on a PC with Corei7-7700 3.60GHz CPU and 32GB of RAM. It spends over 2 minutes to process a 770×530 image, in which 14 seconds are used for geometry modeling, while rest of the time for estimating the light sources' colors. The exact time consumption for light sources' colors calculation is related to the number of light sources, and the more light sources in the scene the more time is required to solve their values. Table 4 shows the time consumptions for solving different number of light sources' colors.

7 Conclusion and Future Work

This paper introduced an automatic framework for recovering the spatially varying illumination of indoor scenes based on a single RGB-D image. We first proposed a method to obtain a rough 3D planar model of a scene's visible and invisible parts based on an input depth map and the assumption that the scene is located in a box with six faces.

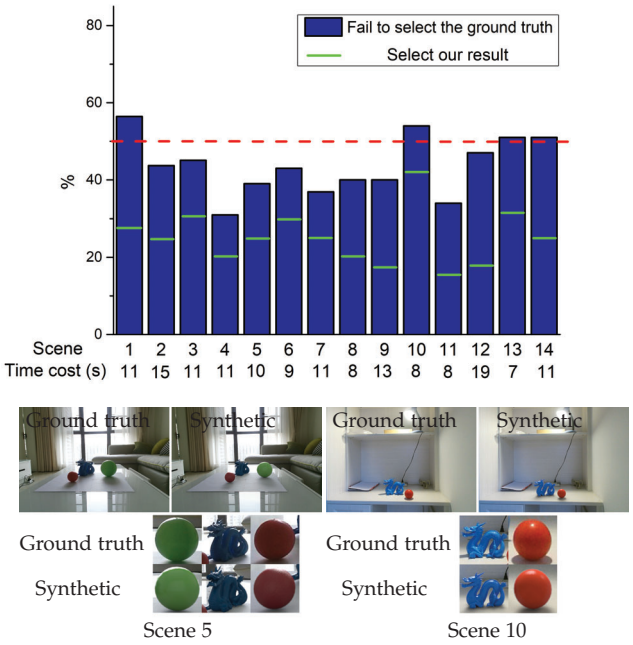
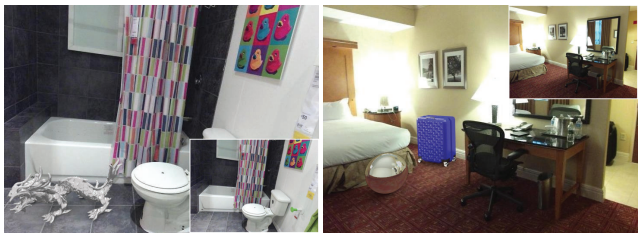


Fig. 16. The results of a user study. For each scene, the blue bar represents the percentage of users who were unable to identify the ground truth, while the green line represents the percentage of users who think our result is more realistic. The horizontal red line indicates the 50% level. The average time required to make a decision for each scene is also demonstrated. Two examples of scenes used in the user study appear below the graph.

Then, we proposed a confidence-scoring based strategy to differentiate and classify light sources and highlights in image. Next, the positions of the visible and invisible light sources are calculated based on the classification result and the geometry model. Spherical harmonics are adopted in an iterative procedure that calculates the intensities of light sources and scene materials based on the recovered scene model. A data-driven method is also proposed to set constraints on light source intensities. An acceleration strategy is employed to make this process more efficient. Using the estimated light sources and geometry model, we generate environment maps at different points in the scene that subsequently support the realistic rendering of virtual objects at these different positions. The experimental results demonstrated the validity and flexibility of our approach.

However, our work still has some limitations. The method cannot process images that contain no light sources or highlights, because such images contain too little information to detect the light sources (e.g., Fig. 17 (a)). Although our method can generate logical environment maps, the appearance of the recovered invisible scene still looks fake, which causes the rendering of objects with strong specular reflection to be unrealistic (e.g., the mirror ball in Fig. 17 (b)). Another limitation is when composting the rendered virtual objects with background image, currently we do not specially matching the blur, ISO etc settings of the input image, causing some inserted objects stick out. In the future, we will utilize the techniques of image restoration to recover and match the settings. Furthermore, the low-quality materials



(a) (b)

Fig. 17. Failure cases. The dragon in the left image does not cast a shadow on the floor, which is not consistent with the surrounding objects. It is easy to see that the mirror ball in the right image is unrealistic.

of the inserted virtual objects make some composting results not completely realistic, which is also a limitation for most of the realistic augmented system. We hope the development of material modeling and illumination estimation join together to enhance the realistic rendering in the future. Last but not least, we will extend our single image illumination estimation method to RGB-D video by adopting additional constrains to keep the temporal stability of the recovered illuminations in our future work.

Acknowledgements

We thank all the anonymous reviewers for their insightful comments and constructive suggestions. This research is supported by National Natural Science Foundation of China (Grant No.61572333,61402081), Open Project Program of the Science and Technology on Optical Radiation Laboratory(Grant No. 61424080109), China National Key Research and Development Plan (Grant No.2016YFB1001200). Ling is supported in part by US National Science Foundation (Grant No.1618398, IIS-1814745 and IIS-1350521). Yanli Liu is the corresponding author.

References

[1] J. Kishimoto, M. Yamaguchi, and K. Koshi, "High dynamic range images capturing with multi-band camera," in *ACM SIGGRAPH 2007 posters*, 2007, p. 64.

[2] A. Darmont, *High Dynamic Range Imaging: Sensors and Architectures*, 1st ed. Washington, D.C.: SPIE, 2012.

[3] J. Kronander, F. Banterle, A. Gardner, E. Miandji, and J. Unger, "Photorealistic rendering of mixed reality scenes," in *Computer Graphics Forum (Proc. Eurographics 2015)*, vol. 34, no. 2, May 2015, pp. 643–665.

[4] P. Debevec, "Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography," in *Proc. SIGGRAPH'98*. Orlando, FL, USA, Jul. 1998, pp. 189–198.

[5] J. Lopez-Moreno, E. Garces, S. Hadap, E. Reinhard, and D. Gutierrez, "Multiple light source estimation in a single image," *Comput. Graph. Forum*, vol. 32, no. 8, pp. 170–182, Dec. 2013.

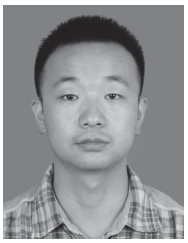
[6] J. Stumpfel, C. Tchou, A. Jones, T. Hawkins, A. Wenger, and P. Debevec, "Direct HDR capture of the sun and sky," in *Proc. AFRIGRAPH '04*, Stellenbosch, South Africa, Nov. 2004, pp. 145–149.

[7] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem, "Rendering synthetic objects into legacy photographs," in *Proc. ACM Siggraph Aisa*, Hong Kong, China, Dec. 2011, pp. 157:1–157:12.

[8] G. Xing, X. Zhou, Q. Peng, Y. Liu, and X. Qin, "Lighting simulation of augmented outdoor scene based on a legacy photograph," *Comput. Graph. Forum*, vol. 32, no. 7, pp. 101–110, Oct. 2013.

[9] T. Yu, H. Wang, N. Ahuja, and W.-C. Chen, "Sparse lumigraph relighting by illumination and reflectance estimation from multi-view images," in *Proc. Siggraph2006*, Boston, MA, USA, Jul. 2006, pp. 175–175.

[10] T. Haber, Fuchs, Christian, P. Bekaer, H.-P. P. Seidel, M. Goesele, and H. P. A. Lensch, "Relighting objects from image collections," in *Proc. CVPR09*. Miami Beach, FL, USA, Jun. 2009, pp. 627–634.
[11] I. Sato, Y. Sato, and K. Ikeuchi, "Illumination from shadows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 3, pp. 290–300, Mar. 2003.
[12] Y. Zhang and Y.-H. Yang, "Multiple illuminant direction detection with application to image synthesis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 8, pp. 915–920, Aug. 2001.
[13] X. Mei, H. Ling, and D. W. Jacobs, "Sparse representation of cast shadows via l1-regularized least squares," in *Proc. ICCV*. Kyoto, Japan, Sep. 2009, pp. 583–590.
[14] E. A. Khan, E. Reinhard, R. W. Fleming, and H. H. Blthoff, "Image-based material editing," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 654–663, 2006.
[15] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth, "Automatic scene inference for 3D object compositing," *ACM Trans. Graph.*, vol. 33, no. 3, p. 32, Jun. 2014.
[16] J. Unger, S. Gustavson, and A. Ynnerman, "Spatially varying image based lighting by light probe sequences: Capture, processing and rendering," *Vis. Comput.*, vol. 23, no. 7, pp. 453–465, Jul. 2007.
[17] J. Unger, A. Wenger, T. C. Hawkins, A. Gardner, and P. Debevec, "Capturing and rendering with incident light fields," 2003, pp. 141–149.
[18] J. Unger, S. Gustavson, P. Larsson, and A. Ynnerman, "Free form incident light fields," *Comput. Graph. Forum (Proc. EGSR 2008)*, vol. 27, no. 4, pp. 1293–1301, 2008.
[19] I. Sato, Y. Sato, and K. Ikeuchi, "Acquiring a radiance distribution to superimpose virtual objects onto a real scene," *IEEE Trans. Vis. Comput. Graph.*, vol. 5, no. 1, pp. 542–547, Mar. 1999.
[20] F. Banterle, M. Callieri, M. Dellepiane, M. Corsini, F. Pellacini, and R. Scopigno, "EnvyDepth: An interface for recovering local natural illumination from environment maps," *Comput. Graph. Forum*, vol. 32, no. 7, pp. 411–420, Oct. 2013.
[21] M. Meilland, C. Barat, and A. Comport, "3D high dynamic range dense visual slam and its application to real-time object relighting," in *ISMAR*, Oct. 2013, pp. 143–152.
[22] K. Hara, K. Nishino, and K. Ikeuchi, "Light source position and reflectance estimation from a single view without the distant illumination assumption," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 493–505, Apr. 2005.
[23] S. Boivin and A. Gagalowicz, "Image-based rendering of diffuse, specular and glossy surfaces from a single image," in *Proc. ACM Siggraph*. Los Angeles, CA, USA, Aug. 2001, pp. 107–116.
[24] R. Ramamoorthi and P. Hanrahan, "On the relationship between radiance and irradiance: Determining the illumination from images of a convex lambertian object," *J. Opt. Soc. Am. A*, vol. 18, no. 10, pp. 2448–2459, Oct. 2001.
[25] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. UIST*, 2011, pp. 559–568.
[26] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *Proc. CVPR*, 2015, pp. 567–576.
[27] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *ECCV*, 2012, pp. 746–760.
[28] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *Proc. 30th IEEE Conf. Comput. Vis. and Pattern Recognition*, 2017, arXiv:1611.08974.
[29] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *Proc. IEEE Conf. Comput. Vis. and Pattern Recognition*, 2013, pp. 564–571.
[30] P. Didyk, R. Mantiuk, M. Hein, and H.-P. Seidel, "Enhancement of bright video features for hdr displays," *Computer Graphics*, vol. 27, no. 4, pp. 1265–1274, 2008.
[31] H.-L. Shen and Q.-Y. Cai, "Simple and efficient method for specularity removal in an image," *Appl. Opt.*, vol. 48, no. 14, pp. 2711–2719, May 2009.
[32] E. H. Land and J. J. McCann, "Lightness and retinex theory," *Journal of the Optical Society of America*, vol. 61, no. 1, pp. 1–11, 1978.
[33] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2004.



Guanyu Xing is an associate researcher in National Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University. He graduated from Zhejiang University with a Ph.D degree in 2013. His research interests are in the area of computer graphics, computer vision and augmented reality.

Yanli Liu is an associate professor in the College of Computer Science, Sichuan University. She graduated from Zhejiang University with a Ph.D degree in 2010. Currently, her research focuses on augmented reality, computer vision and image/video processing.

Haibin Ling is an associate professor in the Center for Data Analytics and Biomedical Informatics, Dept. of Computer & Information Sciences, Temple University. Dr. Ling received Ph.D. from the University of Maryland in Computer Science in 2006. His research interests include computer vision, augmented reality, medical image analysis, visual privacy protection, and human computer interaction.

Xavier Granier is currently a professor in LP2N laboratory, Institut d'Optique Graduate School, Bordeaux-France. Dr. Granier received the Engineer and MS degrees from the Grenoble Institute of Technology and the PhD diploma from University Joseph Fourier in Grenoble, France. His research interests include realistic lighting and expressive rendering, appearance modeling and acquisition.

Yanci Zhang is a Professor in the College of Computer Science of Sichuan University, China. He completed his Bachelors in the Department of Computer Science of Sichuan University, China. He obtained his Ph.D. from Institute of Software, Chinese Academy of Sciences in 2003. His research interests include realtime rendering, virtual reality and parallel computing.