Vision, Modeling, and Visualization (2014)
J. Bender, A. Kuijper, T. von Landesberger, H. Theisel and P. Urban (Eds.)

Interactive Appearance Editing in RGB-D Images

Stephan Bergmann[1], Tobias Ritschel[2], Carsten Dachsbacher[1]

[1]Karlsruhe Institute of Technology [2]Max Planck Institute for Informatics

Abstract

The availability of increasingly powerful and affordable image and depth sensors in conjunction with the necessary processing power creates novel possibilities for more sophisticated and powerful image editing tools. Along these lines we present a method to alter the appearance of objects in RGB-D images by re-shading their surfaces with arbitrary BRDF models and subsurface scattering using the dipole diffusion approximation. To evaluate the incident light for re-shading we combine ray marching using the depth buffer as approximate geometry and environment lighting. The environment map is built from information solely contained in the RGB-D input image exploiting both the reflections on glossy surfaces as well as geometric information. Our CPU/GPU implementation provides interactive feedback to facilitate intuitive editing. We compare and demonstrate our method with rendered images and digital photographs.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Raytracing / Color, shading, shadowing, and texture

1. Introduction

Photo editing has evolved from manipulating individual pixels to complex operations that enable powerful tools for the user to alter and compose photographs that appear genuine. In parallel, sensors have evolved from image sensors with modest resolutions to dozens of mega-pixels. Also depth sensors have recently become affordable and are a valuable addition to 2d image sensors, extending the range of possible editing applications.

We take advantage of the multimodal sensor data and the available processing power to develop interactive high-level image operations. These operations – altering the appearance of objects by modifying the material properties – allow us to alter an image in a physically plausible way. Note that many rendering methods take advantage of the fact that it is not necessary to maintain strict physical correctness to convey a convincing impression due to the properties of the human visual system (e.g. see [FB05]).

Our method takes color images with a matching depth buffer (henceforth referred to as RGB-D images) as input. In this work we focus on changing the appearance of objects in such RGB-D images, especially the impression of an object's glossiness and translucency. One important goal of our work is to avoid elaborate scene and material reconstruction or inverse rendering techniques, and instead maintain a flexible yet easy to implement method.

2. Previous work

There is a substantial body of work concerning the manipulation of digital images so we will limit ourselves to those publications that are most closely related to our work.

Oh et al. [OCDD01] rely on the user to assign depth values to image regions using special user interfaces, and then use this information to enable advanced cloning and relighting operations. Fang and Hart [FH04] make use of shape-from-shading and texture synthesis to replace the texture of objects in digital images. Zelinka et al. [ZFGH05] improve upon this by providing, among other things, interactive feedback of the texture synthesis.

Probably the most similar to our work is that of Khan et al. [KRFB06]: Their method also allows the replacement of an object's BRDF and evaluate the incident light by first employing an inpainting algorithm to fill missing pixels behind an object and then using image pixels to build an environment map (EM) for this purpose. Additionally they provide a method to simulate translucency and transparency of objects. They rely on the "dark-is-deep" assumption to extract depth and normals from the object of interest. Also in contrast to our method, the authors use image warping to build

DOI: 10.2312/vmv.20141269

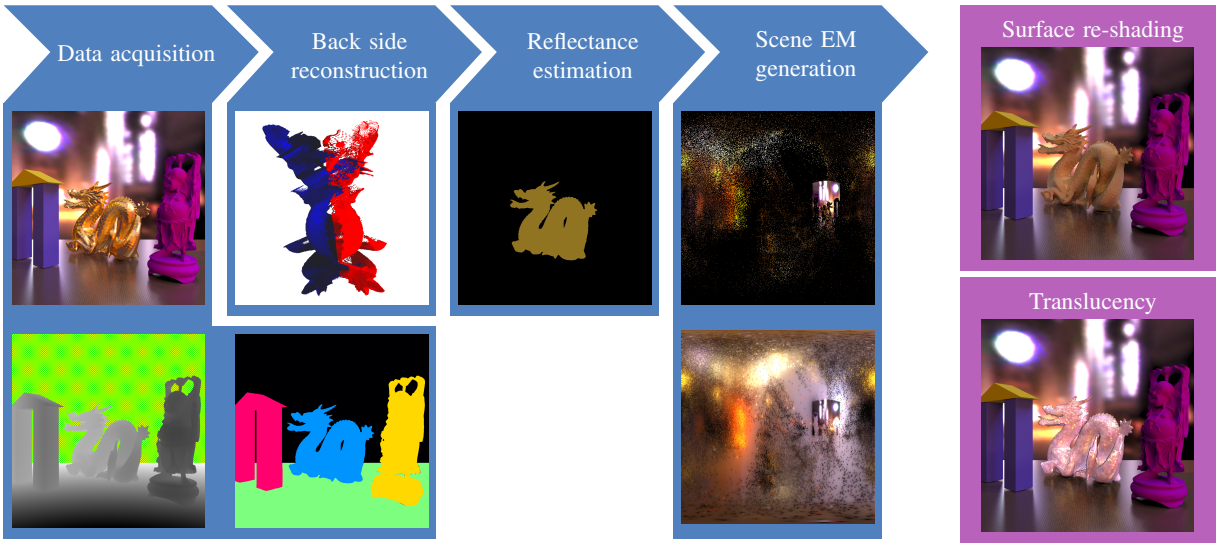

Figure 1: An overview of our method. The blue chevrons symbol preparatory processing steps, the two blocks on the right our main editing steps.The data buffers (synthetic test scene) are connected with their processing step are by the blue background.

the EM and mirror image data to evaluate incident light from directions not directly visible in the image.

Gutierrez et al. [GLMF*08] show how certain properties of the human visual system can be exploited to convincingly add advanced lighting effects like caustics to a digital image.

Not only lighting effects can be added to images but also synthetic objects. To appear credible these object need to be subject to the same lighting conditions as those depicted in the image [Deb98]. Karsch at al. [KHFH11] present a semi-automatic system that relies on a combination of user annotations as well as automatic estimation and refinement steps to generate convincing results.

Changing indirect lighting effects like interreflections is addressed by Caroll et al. [CRA11] who use intrinsic images and decompose the lighting to change the reflectance of objects, taking interreflections into account. In contrast to our work they do not propose evaluating arbitrary BRDFs.

Richardt et al. [RSD*12] also use a combination of 3d and 2d sensors to capture videos that can then be subjected to a variety of effects like relighting or stylistic rendering. To this end they developed a spatio-temporal filtering technique that produces high quality depth maps. When dealing with noisy depth sensor data their filtering methods can be applied as part of the preprocessing step for our algorithm to improve the quality of the results.

3. Interactive Appearance Editing

We will first provide an overview of our technique and describe the steps in our algorithm as outlined in Figure 1. The process can be summarized as follows: We build an environment map (EM) of the scene in the input image which will be used for re-shading surfaces or adding translucency. This EM is needed when sampling incoming light from directions where no geometry is present in the depth buffer.

The processing itself comprises the following steps:

Data acquisition The first step is to acquire and preprocess input data, i.e. a color image with a matching depth buffer. Depending on the method used, a filtering step may be necessary to reduce noise. Additional preprocessing includes generating normals and segmenting the images. Examples of the input data we expect can be seen in Figure 1 beneath the first step.

Back side reconstruction As the depth buffer only samples the frontmost surface, we need to reconstruct the far sides of the visible objects to plausibly re-shade surfaces, which requires computing the incident light from other objects. The reconstructed back sides are stored as a second layer in the depth buffer.

Reflectance estimation The surface reflectance of visible objects is estimated by comparing the incident illumination with the reflected light observed from these objects.

Scene environment map generation We reconstruct an environment map for the scene which will subsequently be used when re-shading objects.

Appearance editing We enable the user to interactively manipulate the objects' material parameters (BRDF, translucency). To update the image, we use the reconstructed environmental lighting as well as ray marching in the two-layer depth buffer to account for local interactions. Note that currently we do not support changing the parameters of an existing translucency. However, we can replace the BRDF and then re-add additional translucency effects.

In order to keep our method simple we made some assumptions which we summarize here: A color and a depth

map are available along with a given segmentation of the image. Segments representing glossy object are marked and we assume Lambertian reflection for the objects in the scene which are not marked as glossy. We discuss some of the consequences of these assumptions in Section 5.

3.1. Data acquisition and pre-processing

There are numerous approaches to acquiring color plus depth images, e.g. time-of-flight cameras, laser scanners combined with a 2D camera, or multi-view photography. There is also the possibility to extract depth information from a 2D image by relying on certain assumptions, e.g. the "dark-is-deep" assumption [LB00, KRFB06] or with user-guided approaches [OCDD01, LGG14].

In particular depth information generated by consumer-grade sensors needs to be filtered before the next processing steps. As our method strives to reconstruct environment lighting from the shading on glossy surfaces, it is sensitive to incorrect or noisy normals. Because of this we filter the depth values before generating normals using a joint bilateral filter [TM98]. Note that more involved pre-processing methods have been proposed, e.g. Yang et al. [YWLZ11] and Han et al. [HLK13]. These might be necessary depending on the quality of the raw data.

For the translucency calculation we use material properties based on physical distance units. To be able to use tabulated or measured material parameters we need either the sensor to directly output physical units or an additional calibration parameter that allows a conversion of sensor data into a known unit.

Along with the filtered depth values and the normals we store world space coordinates and a patch area: We treat each pixel as the projection of a quadrilateral patch and back-project it using the pin-hole camera model and known calibration data of the sensors. We then store the coordinate of the patch center and the surface area of the patch.

For the next steps, we require that the image has been segmented into disjunct segments. The segmentation is assumed to assign individual objects in the scene to different segments, so we will use the terms object and (image) segment interchangeably in the the following descriptions. The problem of segmenting an image is facilitated due to the additional depth information compared to pure color images. A survey on segmentation algorithms can be found in Szeliski's recent book [Sze10]. For simplicity, we used manually created label maps in our examples.

3.2. Back side reconstruction

The sensor depth map obviously does not contain information about the far sides of objects in the image. However, we require at least an approximation thereof in order to include global lighting effects by ray marching. In this step we reconstruct a plausible far side of the image objects. These depth values are stored as a second layer in the depth buffer; the first layer contains the (filtered) sensor depth values.

Image segments are processed individually and a backside is created for each one. Besides a depth offset and a simple mirroring at a plane parallel to the image plane (as used for example in [MES*11]), we implemented a method that does not require the object's symmetry plane to be parallel to the image plane but assumes planar symmetry nonetheless. The basic idea is to select a mirror plane (represented by a normal and a point $\mathbf{r}$ in this plane) by looking at the shape of the object, then optimizing the position of that plane along the plane's normal (see Figure 2):

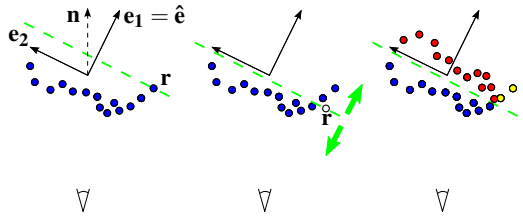

Figure 2: The steps of the backside reconstruction: Selecting a mirror plane normal (left). Optimizing the mirror plane (center). Mirroring the front side;yellow dots are points that are not changed by the mirroring (right).

We regard the segment's set of patch center coordinates P as a point cloud and then perform a principal component analysis (PCA) on this point set, determining the eigenvectors of the covariance matrix. Next, we choose the eigenvector $\hat{\mathbf{e}}$ most similar to the image plane normal $\mathbf{n}$:

$$\hat{\mathbf{e}} = \underset{i\in\{1,2,3\}}{\operatorname{argmin}} |\mathbf{e}_i \cdot \mathbf{n}|,$$

and use $\hat{\mathbf{e}}$ as the normal of the symmetry plane. Initially we chose the point $\mathbf{r}$ from the set P so that $\mathbf{r}$ maximizes $\frac{(\mathbf{r}-\mathbf{c})}{||\mathbf{r}-\mathbf{c}||}\cdot\hat{\mathbf{e}}$ with $\mathbf{r}\in P$, $\mathbf{c}$ being the center of the set P and assuming $\hat{\mathbf{e}}\cdot\mathbf{n} > 0$ (we flip the sign of $\hat{\mathbf{e}}$ otherwise).

In the next step, we optimize the position $\mathbf{r}$ of the assumed symmetry plane by moving it along its normal $\hat{\mathbf{e}}$: For each position we calculate a penalty term $E(\mathbf{r}) = \max(C(\mathbf{r}), O(\mathbf{r}))$, where C is the percentage of points that are behind the mirror plane with respect to the camera position (yellow dots in right image in Figure 2) and O is the percentage of points that would be projected outside the silhouette of the segment when subjected to a pin-hole projection with the camera's calibration data.

Using a simple interval bisection, we search for the minimum of $E(\mathbf{r})$, assuming $E(\mathbf{r})$ to be sufficiently smooth. The search ends when either the change of $E(\mathbf{r})$ from one iteration to the next drops below a threshold or a predefined maximum number of iterations is reached. Then we mirror the front face and the front normals at the chosen plane. The surface color on the back is also assumed to mirror that of the front side. Figure 1 shows a top-down view of the result of this operation for the Stanford Dragon beneath the

second step: The original depth data is shown in blue, the reconstructed second layer in red.

To avoid issues with the ray marching later, we ensure a minimal thickness (the difference between the first and the second depth layer) for each pixel position when performing the mirroring step. This minimal thickness is greater than the ray marching step size and guarantees the we do not miss geometry when ray marching.

3.3. Reflectance estimation

In the step following this one, we will build an environment map of the scene using reflected light on glossy objects. When generating an environment map from colored glossy objects, we need to factor out the color of these objects. To determine this color we compute and compare two directional maps for each of these objects: one containing reflected light from the surface $L_S(\omega)$ and one containing an approximation of the incident light $L_G(\omega)$. We tabulate assumed values of these functions in a environment map-like table but this is no environment map in the usual sense because it may contain already convolved data or violate the distant object assumption.

$L_S(\omega)$ is built from the appearance of the object's surface. For each surface pixel contained in the object's segment, we determine the reflected direction ω_R of the camera ray on the patch center. Next, we perform a ray marching in the depth image from the patch center in the direction ω_R.

If the ray hits geometry we check whether the pixel coordinates of the hit point are inside the current segment. If they are outside (case designated "B" in Figure 3 left) or the ray reaches the depth buffer boundaries without intersecting geometry (cases "A"), we use the color of the ray's originating patch and store it as $L_S(\omega_R)$. We average pixel values in the directional maps in case more than one ray contributes to one direction bin.

If the ray hits geometry inside the same segment (case "C") we have to assume that the object's surface reflects another part of the same object. We discard the ray in this case because we want our directional map to contain one one-bounce reflections of the geometry surrounding the current object.

At this stage the directional map for $L_s(\omega)$ contains only scattered values. To close the gaps, we perform a scattered data interpolation using Shepard's method [She68]. Figure 4 shows the generated map for Stanford dragon in the image shown below the first step in Figure 1.

The second directional map should represent the unconvoluted irradiance $L_G(\omega)$, so we cannot use the surface of the object. To build it we trace rays from the world space center of the object uniformly in all directions (Figure 3 right). When a ray in direction ω_U hits geometry (cases "D") we store the color information of the geometry we hit in $L_G(\omega_U)$, assuming diffuse reflection for this geometry.

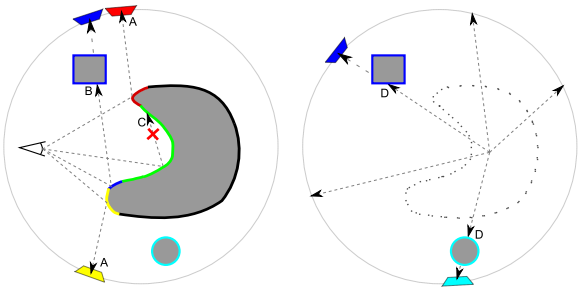

Figure 3: We trace rays from the eye, reflect them at the surface, then store the surface color in the environment map (using the reflection direction) when the ray did not hit the object itself (left). We evaluate incoming light using ray marching from the object center (right).

To leverage all the information present in the image itself, we assume image pixels for which no valid depth information is present (the sensors usually have a maximum range for which they can capture valid depth values) to be distant. Knowing the camera calibration, we can include the color information for these pixels in $L_G(\omega)$, as they likely represent distant geometry that is reflected on the object's surface.

After all pixels have been processed, gaps in $L_G(\omega)$ are again filled by interpolation. Next, we calculate the mean values $\bar{L}_G(\omega)$ and $\bar{L}_S(\omega)$ of both maps, then divide $\bar{L}_G(\omega)$ by $\bar{L}_S(\omega)$ to get an estimate for the reflectance of the surface.

Note that this works and is used only for glossy objects. For diffuse objects this is not needed as we do not use these to build the scene environment map.

3.4. Scene environment map generation

To calculate the shading for given surface point in later steps, we need to be able to evaluate the incoming radiance $L_i(\omega)$ for each direction ω.

To exploit information that might be contained in glossy objects in the scene, we reconstruct an environment map *for the entire scene* from these objects. When shading surfaces, we can then use this environment map in cases where there is no geometry in a direction ω.

We use the segmentation of the image, assuming glossy objects are marked, either by user input or by an additional previous classification step. These glossy segments are now used to build an environment map of the scene.

The method is similar to what we used when generating the $L_S(\omega)$ in the previous section, but here we use only the surface color of pixels whose reflection rays do not intersect *any* geometry (so in the case "B" in Figure 3 nothing will be stored in the EM). The rationale is that the pixels which spawned these rays likely reflect local geometry instead of distant parts of the scene. The surface color has been modulated by the object's reflectance. To undo this, we divide this color by the reflectance we estimated for that object (see section 3.3) before storing it in the EM.

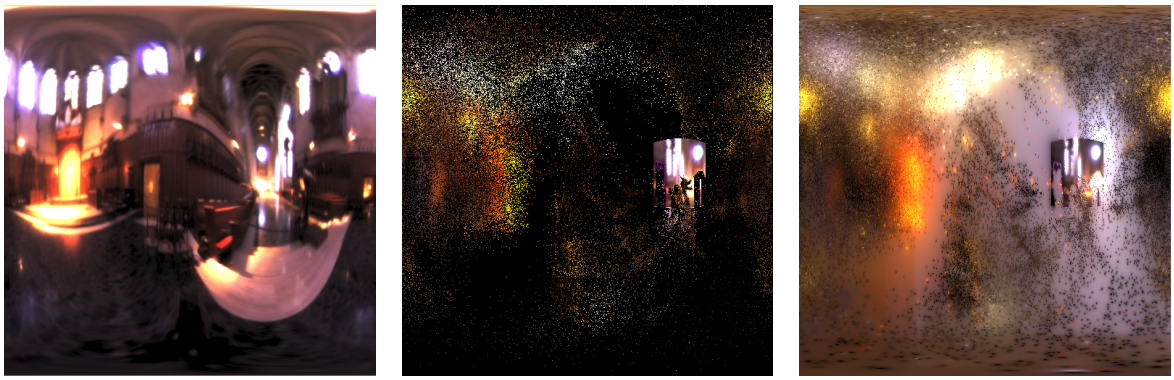

Figure 4: Directional maps (latitude-longitude parameterization): The environment map used for rendering the scene as reference (left); scattered reconstructed values of the directional map $L_S(\omega)$ before interpolation (center) and after interpolation $L_S(\omega)$ (right). The apparent structure on the right side is caused by storing the distant pixel values.

We continue in this fashion until all pixels in the glossy segments have been processed. Depending on the geometry and the chosen resolution of our environment map, multiple rays will contribute to the same entry in the environment map; again we then average these values.

Lastly we add the information from distant image pixels to the environment map (as when generating $L_G(\omega)$) and perform a final interpolation step to obtain the scene EM.

3.5. Surface re-shading

After the preparatory steps, we now focus on our main goal: modifying the appearance of objects in the image. This section describes how we re-shade surfaces using user input and information extracted from the input data.

Re-shading the reflectance (translucency will be discussed in the next section) of an object's surface requires evaluating the well-known reflection integral for all its pixels:

$$L_o(\mathbf{x},\omega_o) = \int_{\Omega^+} f_r(\mathbf{x},\omega_i \rightarrow \omega_o) L_i(\mathbf{x},\omega_i) \cos\theta_i d\omega_i.$$

Here, $\mathbf{x}$ is the world position of the surface patch associated with each pixel, ω_o the view direction and $f_r(.)$ a BRDF. For simplicity we only use the specular component of the Phong model [Pho75] for our example images, however we make no assumptions about the BRDF in general.

The integral is estimated using Monte-Carlo integration with importance sampling and to evaluate $L_i(\mathbf{x},\omega_i)$ we use ray marching in the depth buffer. If a ray originating at $\mathbf{x}$ in direction ω_i hits geometry, we use the value from the color buffer at the hit point. This means we are assuming diffuse reflection for the patch at the hit point. If the ray does not hit any geometry before leaving the depth buffer or reaching a maximum length, we perform a lookup in the generated scene environment map (see Section 3.4) to determine L_i.

The specular coefficient in the Phong model can either be a user parameter or the estimated reflectance value from Section 3.3 can be used which retains the object's color.

In a post-processing step, we filter the obtained values in image space to reduce the variance and blend the new pixel values with the original ones with user-defined weights.

3.6. Translucency rendering

Our system also supports simulating subsurface light transport effects. We follow the technique of Dachsbacher and Stamminger [DS03], using the dipole diffusion approximation [JMLH01]. As in their work, we restrict ourselves to multiple scattering effects to reduce the computational effort knowing that this decision affects the range of materials we can simulate. Although the assumptions of the dipole approximation are not necessarily valid for our application, they yield a plausible image within interactive feedback times. Ultimately we evaluate an integral over the surface on an object using Monte Carlo-Integration:

$$L_o(\mathbf{p},\omega_o) = \frac{1}{\pi} F_t(\omega_o) \frac{\tilde{A}}{|S|} \sum_S \hat{E}(\mathbf{s}_i) R_d(||\mathbf{p}-\mathbf{s}_i||), \quad (1)$$

where F_t is the transmissive Fresnel term, $\tilde{A}$ is an estimate for the total surface area of the object, S is a set of sample points on the object's surface with $\hat{E}(\mathbf{s}_i)$ being the approximated irradiance at sample point $\mathbf{s}_i$, and R_d is the diffuse reflectance [JMLH01]. Simliar to [MES*11] we use a formulation where R_d depends only on the distance between two surface points. Like Jensen and Buhler [JB02] we evaluate the diffusion approximation in two steps, first evaluating the irradiance at all sample points, then iterating over all visible surface points.

We first select a set S of sample points $\mathbf{s}_i$ on the surface of the edited object. We want to sample the surface of the object uniformly, so we need to sample the image pixels proportional to the area of their associated surface patches. To achieve this, we first uniformly generate positions in the image plane (rejecting those outside the segment), then accept samples with a probability proportional to the size of the area of the surface they represent. Note that we need to distribute samples on both front and back side of the object. For each sample point we determine the approximate irradiance $\hat{E}(\mathbf{s}_i)$ by performing a Monte-Carlo integration over

the hemisphere:

$$\hat{E}(s_i) = \frac{\pi}{N} \sum_{j=1}^{N} F_t(\theta_j) L_i(\mathbf{s}_i, \omega_j)$$

The sample directions ω_j are drawn from cosine-weighted distribution over the hemisphere. Marching a ray in the sample direction yields a radiance value $L_i(\mathbf{s}_i, \omega_j)$, either from hitting geometry or accessing the scene EM.

Next, we iterate over all pixels of the object and for each we sum over all the set of samples S evaluating equation 1. $\tilde{A}$ is the sum of the patch areas of the segment's pixels. This sum is doubled to account for the back side.

4. Implementation and Results

We implemented our method using both CPU and GPU processing, using the GPU mainly for performance critical and inherently parallel tasks like interpolation, sampling and filtering. The first steps of our method (up to and including scene EM generation) are performed once per input image and take a few seconds.

One of our goals was to provide interactive feedback to the user when changing material parameters where possible. To keep the feedback latency low, results are updated progressively (for both surface shading and translucency) as a compromise between response times and image quality. While the image quality improves over time, even early outputs allow a user to intuitively grasp the result of an editing operation and to adjust the parameters if necessary. For the Dragon image shown in the paper, the GPU needed 45ms for every 5 rays sampled per surface pixel when re-shading the Dragon. When rendering translucency effects, the GPU runtime was 60ms to update the object for every 10k of surface samples. We collected these performance figures on an Nvidia GTX 760. The image size was 1024^2.

4.1. Synthetic data

To evaluate our method we compared it against images rendered with LuxRender [Lux]. Figure 5 shows an example of a re-shaded surface. The left image (with depth) is used as input data. The image in the middle shows the scene with a re-shaded dragon surface, the Phong exponent has been set to $n = 10$ for this image, the reflectance has been estimated and reused for evaluation. The surface was rendered using 150 samples per pixel. For comparison, we show the same scene rendered with LuxRender with $n = 10$ on the right. Contrary to our approach LuxRender uses multiple bounces so the object's reflectance is more pronounced.

Figure 6 shows that even starting from a diffuse surface we can create the impression of a highly specular material, although the high frequencies of the irradiance function have been smoothed away by the surface's BRDF in the input. As long as there is nearby geometry in the depth buffer, high frequencies created by reflections of this geometry create a plausible impression of a specular surface. However, note the lack of reflected detail in areas where the EM is used to evaluate the incoming light.

The rendering of translucency effects is shown in Figure 7. We used 40k surface samples on the dragon, estimating the irradiance with 100 directions per sample (see [JB02] for a way to relate sample density to surface area and material parameters). The calculated radiance was simply added to the original image values.

4.2. Acquired data

We evaluated different sensors for acquiring RGB-D data. We used the Microsoft Kinect and the Creative Senz3D camera as well as a professional Faro Photon 120 laser scanner. As expected, the quality of the laser scanner's data was superior to that of the other acquisition methods. For the examples we then used the laser scanner's data and applied a joint bilateral filter to the depth values with the RGB data providing the other domain.

One problem that all acquisition methods share is the inability to faithfully scan glossy objects. To remedy this problem we coated these objects with a layer of cyclododecane, rendering their surfaces diffuse for depth acquisition; RGB data was then scanned in a separate pass.

Figure 8 shows results of our method working on acquired data: The left image is the original color data from our sensor combination. In the middle image, the surface of the glossy sphere has been re-shaded to appear less specular. In contrast the right image shows how we changed the appearance of a former diffuse surface, causing it to be perceived as specular.

5. Discussion

The problem of material editing in RGB-D images is inherently hard because of missing or imperfect data, ambiguities and computational requirements. The goal of this work was to provide a comparatively simple yet intuitive method for artistical editing. To achieve this, we had to make a number of assumptions, so naturally there are cases where some of these fail. This can lead to incorrect and/or implausible results. In the following we discuss some of these cases.

No glossy objects in scene When building the environment map, we rely on glossy objects in the scene. If the scene contains no glossy surfaces (or if their normals cover only a small solid angle), a reconstructed EM will not be able to provide a plausible L_i for all directions (see Figure 6 right for an example). This results in missing or wrong data for the final image, when too few or no valid irradiance samples are present. To fill gaps in the image, we experimented with inpainting, e.g. as proposed by Telea [Tel04]. While such methods can easily fill small gaps, they cannot always provide convincing results for larger regions of missing data.

High glossiness assumption We acknowledge that using

Figure 5: Input image with $n = 500$ (left); re-shaded dragon surface with $n = 10$ (center); LuxRender reference image with $n = 10$ (right). Note that the color saturation in the reference image is also higher due to the fact that the global illumination method accounts for multiple reflections, while in our method a material edit affects one bounce only.

Figure 6: A diffuse dragon created from a highly specular input (left). The dragon on the right has been re-shaded using the left image as input and a Phong exponent n=100.

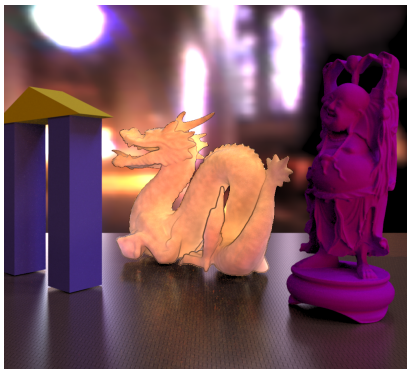

Figure 7: Translucency effect created by first re-shading the dragon to a diffuse appearance and then adding the translucency effect. The silhouette edges are dark due to the Fresnel coefficient when the reconstructed normals are nearly perpendicular to the view direction.

pixel colors directly when building the EM we implicitly assume that the surface is highly glossy because we do not perform a deconvolution. This will cause artifacts if we increase an object's glossiness when re-shading. However, they will only be noticeable if a significant part of an object's surface reflects the environment and not nearby geometry because reflections of nearby geometry will be resolved by ray marching.

Incorrect normals These can result from noisy sensor data or from discontinuities in the input depth image and introduce two problems: During EM construction they cause color data to be splatted into incorrect EM locations and thus whole regions of the EM can be rendered invalid depending on the data density. Second, they can result in obvious errors when shading the surface.

Incorrect back side reconstruction As sensors only capture the front-most surfaces, we make assumptions about the far side of objects in the scene. These assumptions can be wrong, causing the reconstruction back side to be incorrect which in turn can lead to incorrect EMs and incorrectly shaded surfaces. However, our approach yielded satisfying results. Using multiple images with different view points as input data and fusing the information can obviously add more layers to the depth buffer, however, we wanted our work to be applicable to single RGB-D images.

Diffuse object assumption When generating the environment maps and when sampling the irradiance we use the pixel color of the patch hit during ray marching, meaning that we assume a diffuse BRDF. We currently do not use the information about specular surfaces although this information is used for generating the scene environment map. This would allow us to follow further light bounce, however, the input data still lacks all the information about these surfaces.

Indirect lighting effects Our example images (esp. Figure 8 left) clearly show a limitation of our current method: indirect lighting effects are not yet taken into consideration. In future work, we would like to develop methods that allow us to efficiently render plausible indirect lighting changes.

5.1. Future work

Besides those points mentioned in the discussion, we would also like to estimate the properties of the visible surfaces in the scene. We could use estimated parameters for a BRDF model to improve the creation of the environment map. Additionally, this information would also allow us to account for indirect effects more accurately.

We would also like to take into account color information about the far sides of objects that might appear in glossy objects to improve the estimation of back side colors.

Figure 8: Original input image (left) Sphere surface re-shaded with $n = 10$ (center) Bag surface re-shaded with $n = 500$ (right) Note the reflection of the bag's original colors in the sphere and the diffuse reflections on the surface next to the bag.

6. Conclusion

We presented a method to interactively edit material appearance in RGB-D images taking advantage of the depth buffer and using glossy objects in the image to estimate environmental lighting. Together with ray marching against the depth image, this enables plausible surface appearance manipulations and subsurface scattering effects.

Acknowledgments Part of this work has been funded by Software Campus Grant FKZ 01IS12051. Dragon and Happy Buddha model are from the Stanford 3D Scanning Repository. Grace cathedral EM from [Deb98].

References

[CRA11] CARROLL R., RAMAMOORTHI R., AGRAWALA M.: Illumination decomposition for material recoloring with consistent interreflections. *ACM Transactions on Graphics 30*, 4 (2011), 43:1–43:10. 2

[Deb98] DEBEVEC P.: Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH'98* (1998), pp. 189–198. 2, 8

[DS03] DACHSBACHER C., STAMMINGER M.: Translucent shadow maps. In *Proc. Eurographics Workshop on Rendering* (2003), pp. 197–201. 5

[FB05] FLEMING R. W., BÜLTHOFF H. H.: Low-level image cues in the perception of translucent materials. *ACM Transaction on Applied Perception 2*, 3 (2005), 346–382. 1

[FH04] FANG H., HART J.: Textureshop: Texture synthesis as a photograph editing tool. *ACM Transactions on Graphics 23*, 3 (2004), 354–358. 1

[GLMF*08] GUTIERREZ D., LOPEZ-MORENO J., FANDOS J., SERON F., SANCHEZ M., REINHARD E.: Depicting procedural caustics in single images. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia 2008) 27*, 5 (2008), 120:1–120:9. 2

[HLK13] HAN Y., LEE J.-Y., KWEON I. S.: High quality shape from a single RGB-D image under uncalibrated natural illumination. In *ICCV 2013* (2013), vol. 1, pp. 1617–1624. 3

[JB02] JENSEN H. W., BUHLER J.: A rapid hierarchical rendering technique for translucent materials light diffusion in translucent materials. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2002)* (2002). 5, 6

[JMLH01] JENSEN H. W., MARSCHNER S. R., LEVOY M., HANRAHAN P.: A practical model for subsurface light transport. *SIGGRAPH'01* (2001), 511–518. 5

[KHFH11] KARSCH K., HEDAU V., FORSYTH D., HOIEM D.: Rendering synthetic objects into legacy photographs. *ACM Transactions on Graphics 30*, 6 (2011), 157:1–157:12. 2

[KRFB06] KHAN E., REINHARD E., FLEMING R. W., BÜLTHOFF H.: Image-based material editing. *ACM Transactions on Graphics 25*, 3 (2006), 654–663. 1, 3

[LB00] LANGER M. S., BÜLTHOFF H. H.: Depth discrimination from shading under diffuse lighting. *Perception 29*, 6 (2000), 649–660. 3

[LGG14] LOPEZ A., GARCES E., GUTIERREZ D.: Depth from a single image through user interaction. *Spanish Computer Graphics Conference 2014* (2014), 1–10. 3

[Lux] LuxRender GPL physically based renderer. http://www.luxrender.net/. 6

[MES*11] MUNOZ A., ECHEVARRIA J. I., SERON F., LOPEZ-MORENO J., GLENCROSS M., GUTIERREZ D.: BSSRDF estimation from single images. *Computer Graphics Forum (Proc. of Eurographics 2011) 30*, 2 (Eurographics) (2011), 455–464. 3, 5

[OCDD01] OH B. M., CHEN M., DORSEY J., DURAND F.: Image-based modeling and photo editing. *SIGGRAPH'01* (2001), 433–442. 1, 3

[Pho75] PHONG B. T.: Illumination for computer generated pictures. *Commun. ACM 18*, 6 (June 1975), 311–317. 5

[RSD*12] RICHARDT C., STOLL C., DODGSON N. A., SEIDEL H.-P., THEOBALT C.: Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. *Computer Graphics Forum 31*, 2 (2012), 247–256. 2

[She68] SHEPARD D.: A two-dimensional interpolation function for irregularly-spaced data. In *23rd ACM national conference 1968* (1968), pp. 517–524. 4

[Sze10] SZELISKI R.: Segmentation. In *Computer Vision: Algorithms and Applications*. Springer, 2010. 3

[Tel04] TELEA A.: An image inpainting technique based on the fast marching method. *Journal of Graphics Tools 9*, 1 (2004), 25–36. 6

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. *Sixth International Conference on Computer Vision* (1998), 839–846. 3

[YWLZ11] YANG Q.-Q., WANG L.-H., LI D.-X., ZHANG M.: Hierarchical joint bilateral filtering for depth post-processing. *2011 Sixth International Conference on Image and Graphics* (2011), 129–134. 3

[ZFGH05] ZELINKA S., FANG H., GARLAND M., HART J. C.: Interactive material replacement in photographs. In *Graphics Interface* (2005), pp. 227–232. 1