Local Shape Editing at the Compositing Stage

Carlos J. Zubiaga, Gael Guennebaud, Romain Vergne, Pascal Barla

Eurographics Symposium on Rendering - Experimental Ideas & Implementations (2016)
E. Eisemann and E. Fiume (Editors)

Local Shape Editing at the Compositing Stage

Carlos J. Zubiaga†, Gael Guennebaud, Romain Vergne, Pascal Barla
INRIA

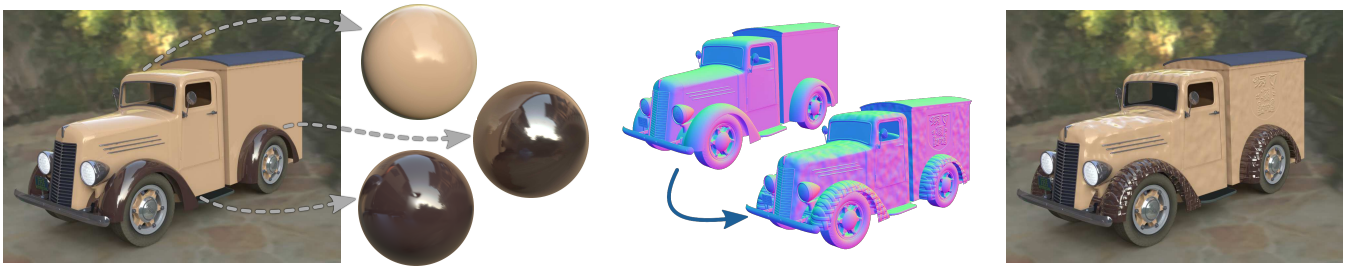


Figure 1: *Our method permits to modify surface shape by making use of the shading and auxiliary buffers output by modern renderers. We first reconstruct shading environments for each object/material combination of the* `Truck` *scene, relying on normal and shading buffers. When normals are then modified by the compositing artist, the color image is recomposited in real-time, enabling interactive exploration. Our method reproduces inter-reflections between objects, as seen when comparing the reconstructed environments for rear and front mudguards.*

Abstract

Modern compositing software permit to linearly recombine different 3D rendered outputs (e.g., diffuse and reflection shading) in post-process, providing for simple but interactive appearance manipulations. Renderers also routinely provide auxiliary buffers (e.g., normals, positions) that may be used to add local light sources or depth-of-field effects at the compositing stage. These methods are attractive both in product design and movie production, as they allow designers and technical directors to test different ideas without having to re-render an entire 3D scene.

We extend this approach to the editing of local shape*: users modify the rendered normal buffer, and our system automatically modifies diffuse and reflection buffers to provide a plausible result. Our method is based on the* reconstruction *of a pair of diffuse and reflection prefiltered environment maps for each distinct object/material appearing in the image. We seamlessly combine the reconstructed buffers in a* recompositing *pipeline that works in real-time on the GPU using arbitrarily modified normals.*

1. Introduction

The image output by a 3D renderer prior to tone mapping may be conceived as the sum of several components (diffuse and reflection shading, emission, transparency, etc). Most modern offline rendering engines output these components in separate image buffers without impeding rendering performance. Artists may then adjust the intensity of each buffer at the compositing stage, before adding them together and applying tone mapping or color grading [Bir05]. Auxiliary components such as position or normal buffers permit additional modifications: adding lights in post-process (using normals) or depth of field (using positions) for instance. This approach is most often preferred to a full re-rendering of the 3D scene that would require considerably larger assets, longer times and different artistic skills. As a result, it is routinely used in product design applications (e.g., Colorway) or in movie production (e.g., Nuke or Flame) to quickly test alternative compositions.

However, any information about lighting is lost past the rendering stage: only shading is available in post-process. Hence if one modifies auxiliary buffers holding 3D normals or positions, it has no effect on shading buffers. One must completely re-render the scene in 3D to take such shape manipulations into account, which is a time-consuming process that forbids interactive exploration.

Our goal in this paper is to grant the editing of *normals* (i.e., local shape) in real-time in post-process, with proper repercussions in diffuse and reflection (either specular or glossy) buffers. An exact reconstruction of these shading buffers would be impossible as we lack much of the necessary 3D data. We instead strive for a plausible result, ensuring that the input diffuse and reflection shading buffers are recovered when reverting to the original normals.

The key idea of our method is to *reconstruct* a pair of pre-filtered

† e-mail:carlos.zubiaga@inria.fr

environments per object/material: one for the diffuse term, the other for the reflection term. Obtaining new shading colors from arbitrarily modified normals then amounts to perform a pair of lookups in the respective prefiltered environment maps. Modifying local shape in real-time then becomes a matter of *recompositing* the reconstructed shading buffers.

Our approach is a first step toward the editing of surface shape (and more generally object appearance) at the compositing stage, which we believe is an important and challenging problem. The rendering and compositing stages are clearly *separate* in practice, involving different artistic skills and types of assets (i.e., 3D scenes vs render buffers). Providing compositing artists with more control over appearance will thus require specific solutions. Our paper makes the following contributions toward this goal (see Figure 1):

- Diffuse and reflection shading environments are *automatically* reconstructed in pre-process for each object/material combinations occurring in the reference rendering (Section 3);
- The reconstructed environments are used to create new shading buffers from *arbitrarily* modified normals, which are recomposited in real time with the reference shading buffers (Section 4).

A simple alternative to our reconstruction approach would be to compute diffuse and reflection shading environments directly at the rendering stage. This could be done for instance by replacing each object/material combination by a sphere with the same material, roughly located at the same position. We have initially tried this approach, but it revealed to be intractable: it is not only a tedious manual process, but also requires multiple costly render passes and does not accurately reproduces inter-reflections between objects.

2. Related work

To the best of our knowledge, the manipulation of appearance at the compositing stage (using both shading and auxiliary buffers) has not been directly addressed in the literature. However, a number of methods are related to our objective, as detailed in the following.

Appearance manipulation The image-based material editing technique of Khan et al. [KRFB06] addresses the difficult problem of manipulating material appearance given a single input image. It mostly relies on an estimation of approximate depth buffer and environment illumination, which are used in a potentially costly re-rendering pass using different materials. Despite the simple estimation heuristics used, results are most often convincing: the human visual system can tolerate quite large discrepancies in physical accuracy [RFWB07]. This is also exemplified in the work of Ritschel et al. [ROTS09] on interactive reflection editing. The method takes inspiration on paintings to manipulate reflections off mirror-like surfaces in non-physical ways. However, it works entirely in 3D space and would not be easily adapted to work at the compositing stage. The surface flows technique [VBFG12] does work exclusively from normal and depth (i.e., auxiliary) buffers. However, the goal of the method is rather to create 3D shape impressions using normal- and depth-based image deformations. Instead, our goal is to manipulate *existing* shading buffers based on modified normals; this is similar to bump mapping [Bli78], but at the compositing stage. As in previous work, our method relies on the tolerance to physical inaccuracies exhibited by human vision.

Lighting reconstruction Another approach would be to reconstruct scene lighting at the compositing stage. Lombardi et al. [LN12] tackle a similar problem: estimating both reflectance and natural illumination from a single image of an object of known shape. The reconstructed lighting environments exhibit artifacts, but these occur only when re-rendering the object with a shinier material compared to the original one. Since material properties can be made available in auxiliary buffers, this method could yield more accurate results in a compositing context. However, as shown by Ramamoorthi et al. [RH01b], there are inherent limitations on the recoverable lighting frequency content depending on the material considered. Most importantly, it is not necessary to decouple material and lighting for local shape editing: recovering a *prefiltered lighting* [HS98] is enough. This approach has been taken by Zubiaga et al. [ZMB*15] to decompose images of shaded spheres (called MatCaps or Lit Spheres [SMGG01]) into diffuse and reflection environment maps. Our work seeks a similar reconstruction, with the added difficulties that object shape is arbitrary (not just a sphere), with multiple objects rendered in global illumination.

2.5D image synthesis Previous work has made use of auxiliary buffers with different goals than local shape editing. A common use is for the filtering of noise resulting from Monte Carlo rendering (e.g., [RMZ13]), as well as upsampling to higher resolutions (e.g., [ZRJ*15]), both methods being employed to reduce rendering costs. Here buffers such as positions, normals and surface IDs are used to select which pixels are likely to reflect similar radiance in a small neighborhood around a pixel of interest. Recent methods such as the work of Tokuyoshi et al. [Tok15] also adapt filtering to material properties. Another use of auxiliary buffers is for novel-view synthesis [LRR*14]: positions, normals and motion flows are used to interpolate between two rendered images. Contrary to these methods, our approach requires to use shading colors in *very large neighborhoods* around the pixel of interest (potentially the whole image) to recover as much shading information as possible.

3. Reconstruction

In this paper, we focus on modifying the apparent shape of *opaque* objects at the compositing stage. We thus only consider the diffuse and reflection buffers, the latter resulting from reflections off either specular or glossy objects. These shading buffers exhibit different frequency content: diffuse shading is low-frequency, while reflection shading might contain arbitrarily high frequencies. As a result we use separate reconstruction techniques for each of them.

Both techniques take their reference shading buffer D_0 or R_0 as input, along with an auxiliary normal buffer $\mathbf{n}$. Reconstruction is performed separately for each surface patch $\mathcal{P}$ belonging to a same object with a same material, and identified thanks to a surface ID buffer. We also take as input ambient and reflection occlusion buffers α_D and α_R, which identify diffuse and specular visibility respectively (see Appendix). The latter is used in the reconstruction of the reflection environment. We use Modo to export all necessary shading and auxiliary buffers, as well as camera data. The normals are transformed from world to screen-space prior to reconstruction; hence the diffuse and reflection environments output by our reconstruction techniques are also expressed in screen space.

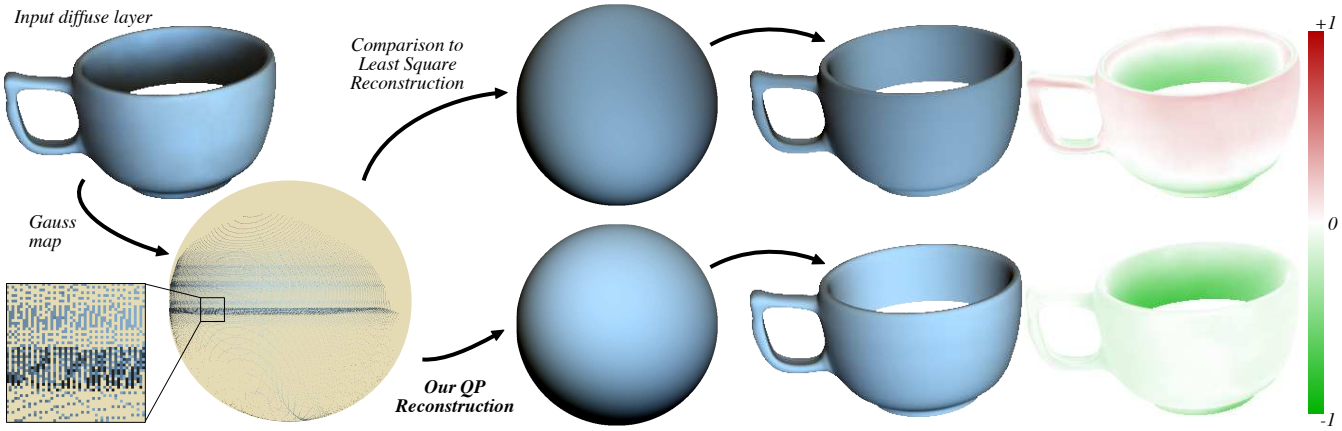


Figure 2: *Reconstruction of the prefiltered diffuse environment map. From left to right: the pixels of the input diffuse layer of the selected object are scattered inside the Gauss map. This shading information is then approximated by low-order spherical harmonics using either a Least Square fit (top) or our Quadratic Programming formulation (bottom) leading to the respective reconstructed environment maps. To evaluate the reconstruction quality, those environment maps are then applied to the original objects, and the signed residual is shown using a color code (shown at right). Observe how our QP reconstruction guarantees a negative residual.*

3.1. Diffuse environment

In this section our goal is to reconstruct a complete prefiltered diffuse environment $D : S^2 \to \mathbb{R}^3$, parametrized by surface normals $\mathbf{n}$. The map D should match the input diffuse buffer inside $\mathcal{P}$: for each selected pixel $\mathbf{x} \in \mathcal{P}$, $D(\mathbf{n}(\mathbf{x}))$ should be as close as possible to $D_0(\mathbf{x})$. However, as illustrated by the Gauss map in Figure 2, this problem is highly ill-posed without additional prior. First, it is under-constrained in regions of the unit sphere *not covered* by the Gauss map of the imaged surface. This is due to sharp edges, occluded regions and highly curved features producing a very sparse sampling. Second, the problem might also be over-constrained in areas of the Gauss map covered by *multiple sheets* of the imaged surface, as they might exhibit different diffuse shading values due to global illumination (mainly occlusions).

Since diffuse shading exhibits very low frequencies, we address the coverage issue by representing D with low-order Spherical Harmonics (SH) basis functions, which have the double advantage of being globally supported and of exhibiting very good extrapolation capabilities. We classically use order-2 SH, only requiring 9 coefficients per color channel [RH01a]. The reconstructed prefiltered diffuse environment is thus expressed by:

$$D(\mathbf{n}) \;=\; \sum_{l=0}^{2} \sum_{m=-l}^{l} c_{l,m} Y_{l,m}(\mathbf{n}). \qquad (1)$$

The multiple-sheets issue is addressed by reconstructing a prefiltered environment as if no occlusion were present. This choice will facilitate shading edition as the local residual $D_0(\mathbf{x}) - D(\mathbf{n}(\mathbf{x}))$ is then directly correlated to the amount of local occlusion. Formally, it amounts to the following constrained quadradic minimization problem:

$$c^{\star}_{l,m} \;=\; \arg\min_{c_{l,m}} \sum_{\mathbf{x}\in\mathcal{P}} \|D_0(\mathbf{x}) - D(\mathbf{n}(\mathbf{x}))\|^2,$$
$$s.t. \qquad D(\mathbf{n}(\mathbf{x})) \geq D_0(\mathbf{x}),$$

which essentially says that the reconstruction should be as close as possible to the input while enforcing negative residuals. This is a standard Quadratic Programming (QP) problem that we efficiently solve using a dual iterative method [GI83].

Figure 2 compares our approach to a standard least squares (LS) reconstruction. As made clear in the two right-most columns, our QP method produces shading results more plausible than the LS method: residuals are negative by construction and essentially correspond to darkening by occlusion.

Validation The left column of Figure 6 shows reconstructed diffuse shading for a pair of environment illuminations. We project the light probes onto the SH basis and use them to render a 3D teapot model, from which we reconstruct the diffuse environments. In this case, there is no occlusion, only direct lighting: our reconstruction exhibits very good results as shown by the difference images with the ground truth environments.

3.2. Reflection renvironment

At first glance, the problem of reconstructing a prefiltered reflection environment map is similar to the diffuse case. Our goal is to reconstruct a complete prefiltered reflection environment $R : S^2 \to \mathbb{R}^3$ parametrized by the reflected view vector $\mathbf{r} = \text{reflect}(\mathbf{v}, \mathbf{n})$, where $\mathbf{v}$ and $\mathbf{n}$ are the local view and normal vectors respectively. As before, R should match the input reflection buffer: for each selected pixel $\mathbf{x} \in \mathcal{P}$, $R(\mathbf{r}(\mathbf{x}))$ should be as close as possible to $R_0(\mathbf{x})$.

On the other hand, the reflection buffer contains arbitrarily high-frequencies, which prohibits the use of a SH representation. We thus propose to represent and store R in a high resolution dual-paraboloid map [HS98] that we fill in three steps:

1. Mapping from R_0 to R while dealing with overlapping sheets;
2. Hole-filling of R using a spherical harmonic interpolation;
3. Smoothing of the remaining discontinuities.

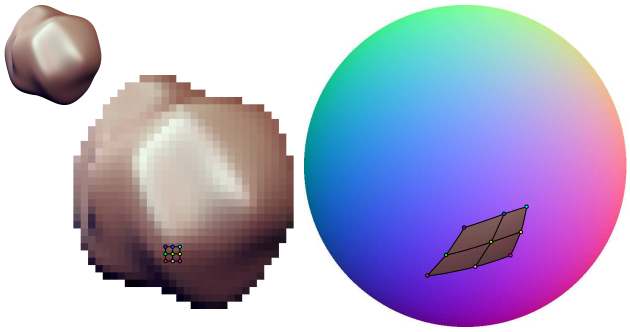


Figure 3: *A 3×3 pixel neighborhood (shown at left on a low-resolution image for clarity) is mapped to four contiguous spherical quads on the Gauss sphere of rectified normals (right). The color inside each spherical quad is computed by bilinear interpolation inside the input image using spherical barycentric coordinates.*

Partial reconstruction

For the reconstruction of the diffuse map, interpreting each input pixel as a simple point sample proved to be sufficient. However, in order to reconstruct a sharp reflection map, it becomes crucial to fill the gaps between the samples with as much accuracy as possible. To this end, we partition the input set of pixels into smooth patches according to normal and depth discontinuities. Each smooth patch has thus a continuous (i.e., hole-free) image once mapped on S^2 through the reflected directions $\mathbf{r}$. Due to the regular structure of the input pixel grid, the image of the patch is composed of adjacent spherical quads and triangles (at patch boundaries). This is depicted in Figure 3 for a block of 3×3 pixels.

Depending on object shape and camera settings, each patch may self-overlap, and the different patches can also overlap each other. In other words, a given reflection direction $\mathbf{r}$ might coincide with several polygons. We combine shading information coming from these different image locations using two types of weights. First, we take as input an auxiliary buffer α_R storing *reflection occlusions*. Examples are shown in the right of Figure 7 and in the Appendix. This information tells us whether the majority of reflected rays used to compute the shading value come from the environment (i.e., other objects). This buffer can be treated as a binary mask to discard polygons exhibiting self-reflections as those polygons are unreliable to reconstruct the environment. Second, among the remaining polygons, we favor the ones that have a small extent on the unit sphere; they correspond to image locations where the object is the least curved, and hence provide the most accurate shading information. Formally, we compute a weight $w_k = (1 - \cos^{-1}(\ell_k)/\pi)^\eta$ for each polygon k, with ℓ_k the longest arc defined by any pair of its vertices. We use $\eta = 200$, which has the effect of selecting the smallest polygons with only a slight blend between nearly equal-sized ones.

A partial reflection environment is thus obtained by combining recovered shading values for any query direction $\mathbf{r}$:

$$R(\mathbf{r}) = \frac{\sum_{k=1}^{N(\mathbf{r})} w_k \sum_{j \in Q_k} \lambda_j^k(\mathbf{r}) R_0(\mathbf{x}_j)}{\sum_{k=1}^{N(\mathbf{r})} w_k}, \qquad (2)$$

where $N(\mathbf{r})$ is the number of *unoccluded* spherical polygons containing $\mathbf{r}$, Q_k is the set of corner indices of the k-th polygon, and λ_j^k are barycentric coordinates enabling the interpolation of the shading colors inside the k-th polygon. For polygons embedded on a sphere, barycentric coordinates can be computed as described in Equation 8 of [LBS06]. We use normalized spherical coordinates, which amounts to favor the partition of unity property over the linear precision property on the sphere (i.e., Equation 2 instead of 3 in [LBS06]).

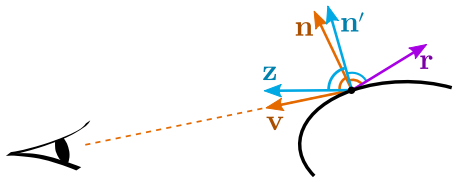


Figure 4: *A normal $\mathbf{n}$ is "rectified" to $\mathbf{n}'$ prior to reconstruction. The reflection of the view vector $\mathbf{v}$ is then given by $\mathbf{r} = reflect(\mathbf{z}, \mathbf{n}')$.*

In order to quickly recover the set of spherical polygons containing $\mathbf{r}$, we propose to first warp the set S^2 of reflected directions to a single hemisphere so that the search space can be more easily indexed. To this end, we compute a *rectified normal* buffer $\mathbf{n}'$ such that $\mathbf{r} = \text{reflect}(\mathbf{z}, \mathbf{n}')$, where $\mathbf{z} = (0,0,1)^T$ as shown in Figure 4. This is obtained by the bijection $\mathbf{n}'(\mathbf{r}) = \frac{\mathbf{r}+\mathbf{z}}{\|\mathbf{r}+\mathbf{z}\|}$. In a preprocess, we build a 2D grid upon an orthographic projection of the Gauss map of these rectified screen-space normals. For each polygon corresponding to four or three connected pixels, we add its index in the cells it intersects. The intersection is carried out conservatively by computing the 2D convex hull of the projected spherical polygon. Then, for each query reflection vector $\mathbf{r}$, we compute the spherical barycentric coordinates λ_j^k of each polygon of index k in the cell containing $\mathbf{n}'(\mathbf{r})$, and pickup the polygons having all λ_j^k positive. In our implementation, for the sake of simplicity and consistency with our 2D grid construction, we compute spherical barycentric coordinates with respect to the rectified normals $\mathbf{n}'$, for both intersections and shading interpolation (Equation 2).

Hole-filling and regularization

Evaluating Equation 2 for each direction $\mathbf{r}$ in a dual-paraboloid map yields a partial reconstruction, with holes in regions not covered by a single polygon and discontinuities at the transition between different smooth parts.

For instance, the bright red region in the top row of Figure 5 correspond to reflection directions where no shading information is available (i.e., $N(\mathbf{r}) = 0$). It is necessary to fill these empty regions to guarantee that shading information is available for all possible surface orientations. In practice, we perform a harmonic interpolation directly on a densely tessellated 3D sphere, with vertices indexed by $\mathbf{r}$ matching exactly the pixels of the dual-paraboloid map. The tessellation is thus perfectly regular except at the junction between the front and back hemispheres. We use a standard Finite Element Discretization with linear basis functions over a triangular mesh to solve the Laplacian differential equation, while setting shading values recovered by Equation 2 as Dirichlet boundary constraints [IGG*14].

A result is shown in the middle row of Figure 5: holes are prop-

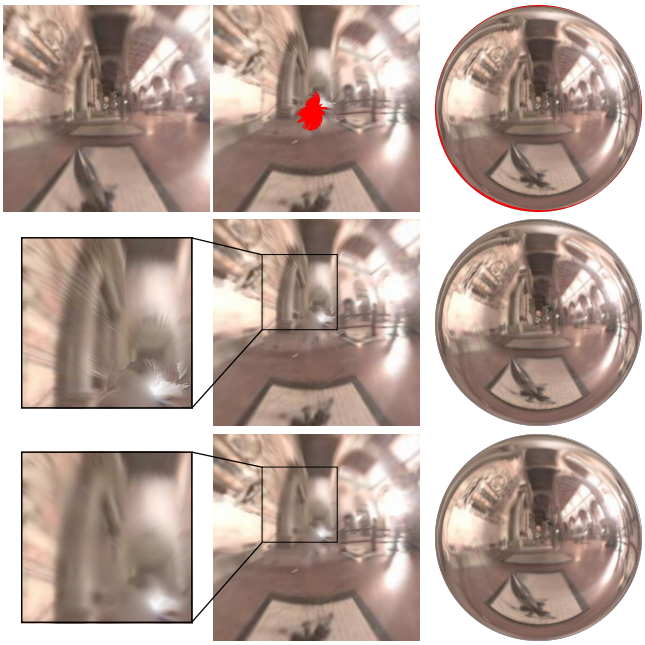


Figure 5: *A dual paraboloid map reconstructed with our approach is shown in the top row: it only partially covers the space of reflected view vectors. The missing information, shown in red, appears confined to the silhouette when visualized with a LitSphere at right. We focus on the back part of the paraboloid map in the middle row to show the result of the hole-filling procedure. Missing regions are completed, but some discontinuities remain; they are corrected by our regularization pass as shown in the bottom row.*

Figure 6: *Reconstruction results for diffuse (left column) and reflection (right column) shading, using* `Galileo` *(top row) and* `rnl` *(bottom row) light probes. Each quadrant shows the reference prefiltered environment, along with the rendering of a 3D teapot assuming distant lighting and no shadowing or inter-reflections. The teapot image is then used to reconstruct the prefiltered environment using either the method of Section 3.1 or 3.2, and a boosted absolute color difference with the reference environment is shown.*

erly filled in, but some shading discontinuities remain. Those are caused by spatial discontinuities in Equation 2 occurring when spatially disconnected polygons are used for neighboring directions in the environment. We thus apply a last post-processing step where we slightly blur the environment along those discontinuities. We identify them by computing a second dual-paraboloid map storing the 2D image position of the polygon that contributed the respective shading color. This map is simply obtained by replacing the shading values $R_0(\mathbf{x}_j)$ in Equation 2 by the 2D coordinates $\mathbf{x}_j$. We then compute the gradient of these maps and use its magnitude to drive a spatially-varying Gaussian blur. The effect is to smooth the radiance on discontinuities caused by two or more remote polygons projected next to one another. An example of regularized environment is shown in the bottom row of Figure 5.

Visualization Dual paraboloid maps are only used for practical storage purposes in our approach. It should be noted that once applied to a 3D object, most of the shading information in the back part of the map gets confined to the silhouette, as shown in the right column of Figure 5. In the following, we thus prefer to use shaded 3D spheres seen in orthographic projection (i.e., Lit Spheres [SMGG01]) to visualize the reconstructed shading environments (both diffuse and reflections). In practice, perspective projection only lets appear a subset of filled-in shading values close to object contours.

Validation The right column of Figure 6 shows reconstruction results for a pair of known environment lighting. As before, a 3D teapot is rendered using the light probe, and then used for reconstructing reflection environment maps. The difference between our reconstruction and the ground truth is small enough to use it for shape editing.

4. Recompositing

The outcome of the reconstruction process is a set of SH coefficients and dual paraboloid maps for all object/material combinations appearing in the image. Obtaining reconstructed shading buffers simply amounts to evaluate shading in the appropriate SH basis or environment map, using arbitrary screen-space normals. A benefit of this approach is that any normal manipulation may then be used in post-process; we give some practical examples in Section 5.

However, we must also ensure that the independently reconstructed shading buffers are seamlessly recombined in the final image. In particular, when a normal is left untouched by the compositing artist, we must guarantee that we reproduce the reference diffuse and reflection shading colors exactly. This is the goal of the recompositing process: taking as input an arbitrarily modified normal buffer, it combines the reconstructed prefiltered environments

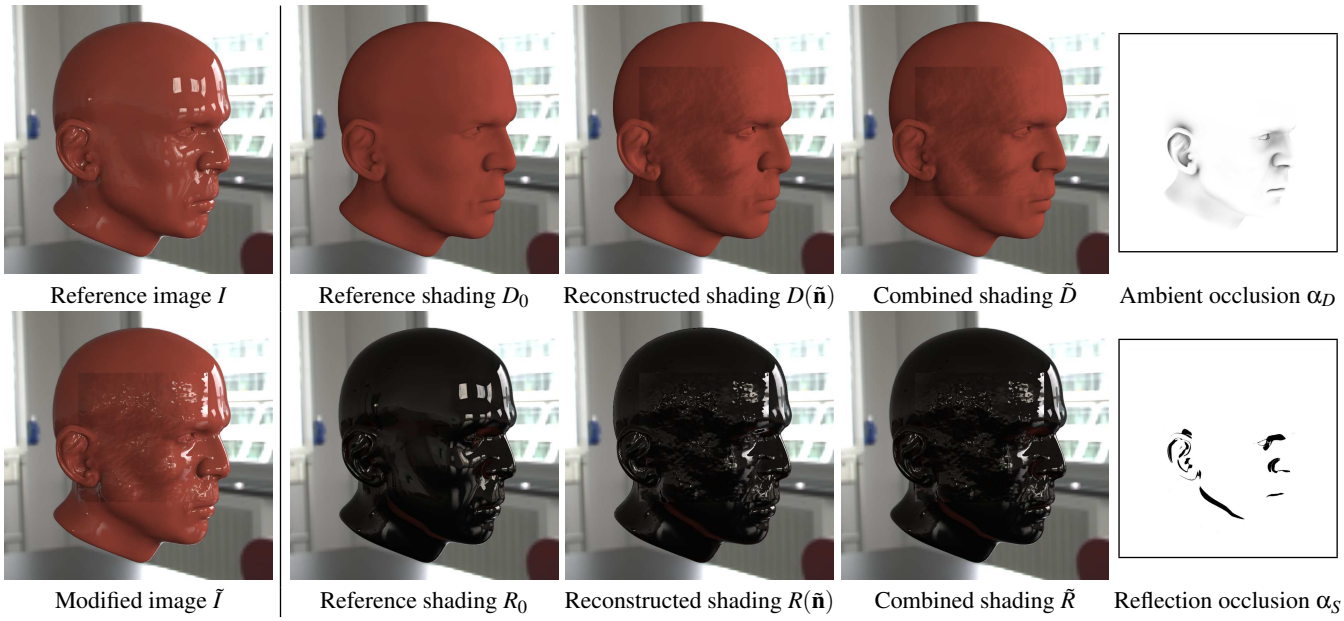


Figure 7: *Illustration of the different terms involved in recompositing (Equations 3-5). The left column shows reference and modified images, where we have perturbed the normal buffer with a 2D perlin noise in a square region. This permits to show the behavior of our approach on both modified & unmodified portions of the image (i.e., inside & outside the square); in the latter case, $\tilde{I} = I$. The remaining columns present the diffuse (top) and reflection (bottom) terms. Starting with the reference shading at left, we then show reconstructed shading using prefiltered environments. Reference and reconstructed shading terms slightly differ in some unmodified regions (e.g., inside the ear). The combined shading terms correct these effects with the aid of ambient and reflection occlusion buffers, shown in the rightmost column.*

with rendered input buffers to produce a final color image where the apparent shape of objects has been altered. It works in parallel on all pixels; hence we drop the dependence on $\mathbf{x}$ for brevity.

Combined diffuse term Given a modified normal $\tilde{\mathbf{n}}$, we define the combined diffuse term $\tilde{D}$ by:

$$\tilde{D} \;=\; \alpha_D \left\lfloor \underbrace{D_0 - D(\mathbf{n})}_{\text{residual}} + D(\tilde{\mathbf{n}}) \right\rfloor + (1-\alpha_D)\, D_0, \tag{3}$$

where the ambient occlusion term α_D is used to linearly interpolate between the reference and reconstructed diffuse colors. The rationale is that highly occluded areas should be preserved to prevent the introduction of unnatural shading variations.

The $D_0 - D(\mathbf{n})$ term is used to re-introduce residual differences between the reference and reconstructed buffers. It corresponds to local darkening of diffuse shading that could not be captured by our global reconstruction. The $\lfloor \cdot \rfloor$ symbol denotes clamping to 0, which is necessary to avoid negative diffuse shading values. This is still preferable to a multiplicative residual term $D_0/D(\mathbf{n})$ as it would raise numerical issues when $D(\mathbf{n}) \approx 0$. Observe that if $\mathbf{n} = \tilde{\mathbf{n}}$ then $\tilde{D} = D_0$: the reference diffuse shading is exactly recovered when the normal is not modified.

Combined reflection term Contrary to the diffuse case, we cannot apply the residual approach between the reference and reconstructed reflection buffers as it would create ghosting artifacts. This is because reflections are not bound to low-frequencies as in the diffuse case. Instead, given a modified normal $\tilde{\mathbf{n}}$ and a corresponding modified reflection vector $\tilde{\mathbf{r}} = \text{reflect}(\mathbf{v}, \tilde{\mathbf{n}})$, we define the combined reflection term $\tilde{R}$ by:

$$\tilde{R} \;=\; \nu_{\mathbf{r},\tilde{\mathbf{r}}}\, \alpha_R\, R(\tilde{\mathbf{r}}) + (1 - \nu_{\mathbf{r},\tilde{\mathbf{r}}}\, \alpha_R)\, R_0, \tag{4}$$

where $\nu_{\mathbf{r},\tilde{\mathbf{r}}} = \min(1, \cos^{-1}(\mathbf{r}\cdot\tilde{\mathbf{r}})/\varepsilon)$ computes the misalignment between original and modified reflection vectors (we use $\varepsilon = 0.005\pi$), and α_R is the reflection occlusion term. The latter serves the same purpose as the ambient occlusion term for the diffuse case. Equation 4 performs a linear interpolation between reference and reconstructed reflection colors based on $\nu_{\mathbf{r},\tilde{\mathbf{r}}}\, \alpha_R$. As a result, if $\mathbf{n} = \tilde{\mathbf{n}}$, then $\nu_{\mathbf{r},\tilde{\mathbf{r}}} = 0$ and $\tilde{R} = R_0$: the reference reflection shading is exactly recovered when the normal is left unmodified.

Final composition The final image intensity $\tilde{I}$ is given by:

$$\tilde{I} \;=\; \alpha \left(k_D \tilde{D} + k_R \tilde{R}\right)^{\frac{1}{\gamma}} + (1-\alpha)\, I, \tag{5}$$

where the diffuse and reflection coefficients k_D and k_R are used to edit the corresponding shading term contributions ($k_D = k_R = 1$ is the default), γ is used for gamma correction (we use $\gamma = 2.2$ in all our results), and α identifies the pixels pertaining to the background (e.g., showing an environment map), which are already gamma corrected in our input color image I.

Equation 5 is carried out on all color channels separately. Figure 7 shows an example of the recompositing process on a simple scene holding a single object, where input normals have been corrupted by a 2D Perlin noise restricted to a square region. The final colors using original and modified normals are shown in the left-

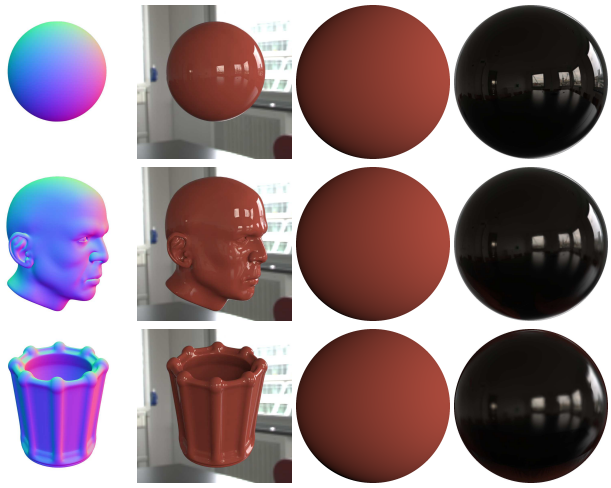


Figure 8: Reconstructed *diffuse and reflection prefiltered environment maps for three objects: the red sphere (top), head (middle) and vase (bottom) models. The diffuse environment reconstructions are nearly identical in all three cases. The sharpness of the reflection environment reconstructions vary with object shape.*

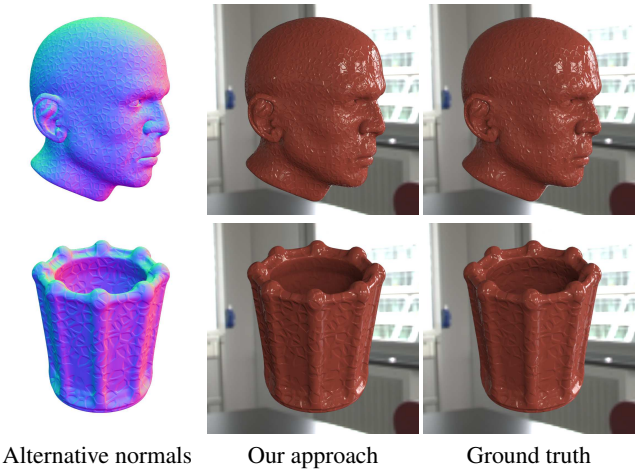


Figure 9: *A Voronoi-based bump texture is mapped onto the red head and vase models in 3D, yielding an alternative normal buffer. Our approach is used to modify shading at the compositing stage in real-time, with a resulting appearance similar to re-rendering the object using the bump map (ground truth).*

most column; the remaining columns show the different gamma-corrected shading terms. The top row focuses on the diffuse term ($k_R = 0$), while the bottom row focuses on the reflection term ($k_D = 0$). The importance of recombining reconstructed and reference diffuse shading done in Equation 3 becomes apparent when comparing $D(\tilde{\mathbf{n}})$ and $\tilde{D}$. In particular, it permits to seamlessly reproduce D_0 outside of the square region (e.g., inside the ear). Similarly, using Equation 4 permits to remove implausible bright reflections in reflection shading (e.g., inside the ear or below the eyebrow).

5. Experimental results

We have implemented the recompositing process of Section 4 in Gratin [VB15], a programmable node-based system working on the GPU. It permits to test various normal modification algorithms in 2D by programming them directly in GLSL, while observing results in real-time as demonstrated in the supplemental video. Alternatively, normal variations can be mapped onto 3D objects and rendered as additional auxiliary buffers at a negligible fraction of the total rendering time. It then grants compositing artists the ability to test and combine different variants of local shape details in post-process. We demonstrate both the 2D and 3D normal editing techniques in a set of test scenes rendered in global illumination.

We start with three simple 3D scenes, each showing a different object with a same material in a same environment illumination. The normal buffer and global illumination rendering for each of these scenes are shown in the first two columns of Figure 8. Diffuse and reflection environments are reconstructed from these input images and shown in the last two columns, using Lit Spheres. The reconstructed diffuse environments are nearly identical for all three objects. However, the quality of reconstruction for the reflection environment depends on object shape. The sphere object serves as a reference and only differs from the LitSphere view due to perspective projection. With increasing shape complexity, in particular when highly curved object features are available, the reflection environment becomes less sharp. However, this is usually not an issue when we apply the reconstructed environment to the same object, as shown in Figures 9 and 10.

We evaluate the visual quality of our approach on the head and vase objects in Figure 9. The alternative normal buffer is obtained by applying a Voronoi-based bump map on the object in 3D. We use the reconstructed environments of Figure 8 and our recompositing pipeline to modify shading buffers in the middle column. The result is visually similar to a re-rendering of the scene using the additional bump map, shown in the right column. A clear benefit of our approach is that it runs in real-time independently of the rendered scene complexity. In contrast, re-rendering takes from several seconds to several minutes depending on the scene complexity.

Figure 10 demonstrates three interactive local shape editing tools that act on a normal $\mathbf{n} = (n_x, n_y, n_z)$. Normal embossing is inspired from the LUMO system [Joh02]: it replaces the n_z coordinate with βn_z where $\beta \in (0, 1]$ and renormalize the result to make the surface appear to "bulge". Bump mapping perturbs the normal buffer with an arbitrary height map, here a fading 2D ripple pattern (the same manipulation is applied in Figure 7 with a 2D Perlin noise). Bilateral smoothing works on projected normals $\bar{\mathbf{n}} = (n_x, n_y)$ using an auxiliary depth buffer to preserve object contours.

In more complex scenes, images of objects may appear in the reflections of each other. This is what occurs in Figure 11, which shows a `Table top` scene with various objects: a cup made of porcelain with a metal spoon, a reddish coated kettle with an aluminum handle, and a vase made of a glossy material exhibiting blurry reflections. Despite the increased complexity, our method still produces a plausible result when normals are modified with a

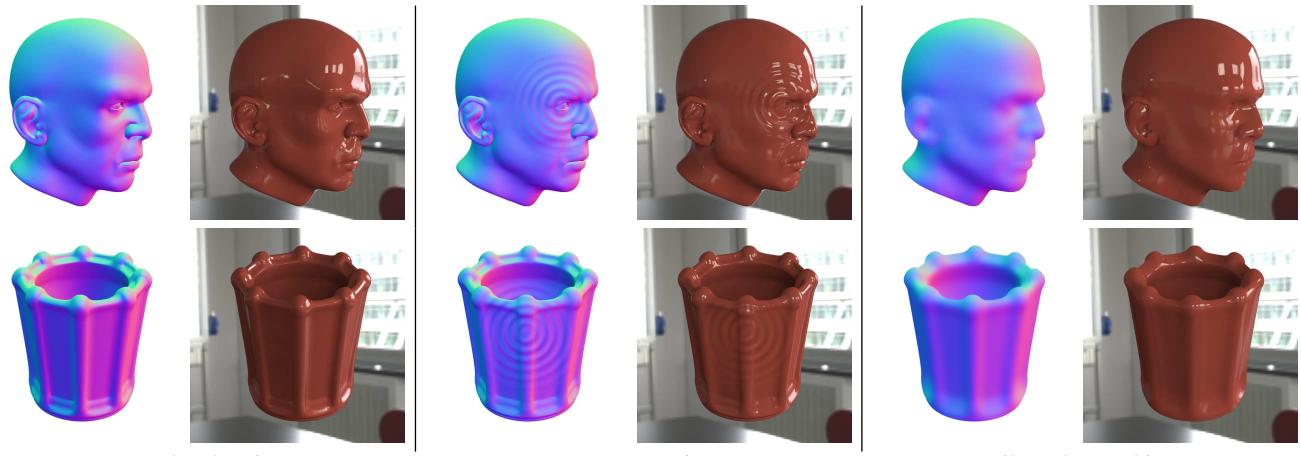


Figure 10: *Our technique permits to apply arbitrary modifications to the normal buffer, while still yielding plausible shading results. Normal embossing is applied to the eyebrow of the head, and the whole vase, resulting in an apparent bulging of local shape. Any bump texture may be applied as a decal to modify normals: we use a 2D fading ripple pattern, affecting both diffuse and reflection shading. Local shape details may also be removed: we apply a cross bilateral filter on normals, using an auxiliary depth buffer to preserve occluding contours.*

Model	Diffuse	Reflection	
	rec.	Partial rec.	Hole filling
Red sphere	110	315	215
Red head	184	470	280
Red vase	147	385	192
Cup	29	45	325
Kettle	65	97	650
Black vase	64	133	4000
Truck body	119	120	6000
Front mudguard	25	45	2500
Rear mudguard	14	26	3700

Table 1: *Timings in ms for each reconstruction stage.*

noise texture on the cup, an embossed symbol on the kettle body and a carved pattern on the vase. The results remain plausible even when the material properties are edited, as shown in the right column where we decrease the diffuse intensity and increase the specular intensity. The reconstructed diffuse and reflection environments are shown combined in Figure 12, before and after material editing has been performed. Observe in particular how the reflections of nearby objects have been properly reconstructed. The reflected window appears stretched in the cup environment. This is because it maps to the highly curved rim of the cup. However, when reapplied to the same object, stretching goes unnoticed.

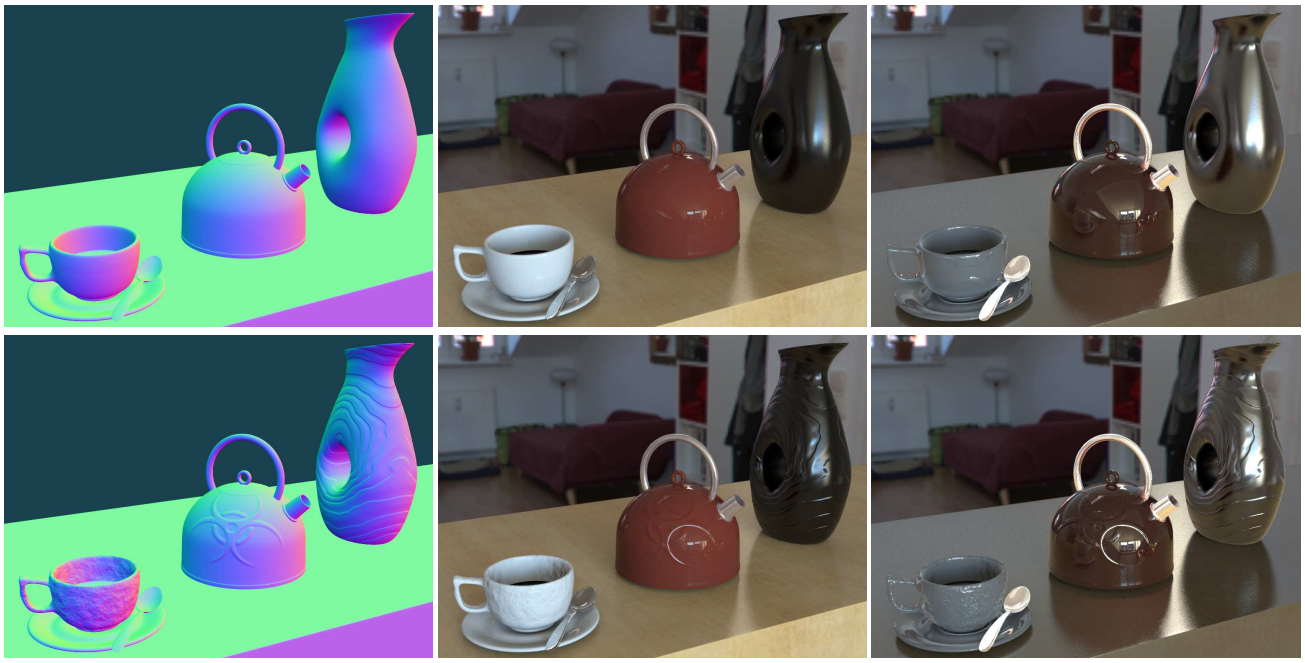
The `Truck` scene of Figure 1 is more challenging: not only object parts are in contact, but each cover a relatively small subset of surface orientations. Nevertheless, as shown in the reconstructed shading environments, our method manages to capture a convincing appearance that reproduces inter-reflections between different parts. This permits to generate a plausible result when normals are modified to apply a noise texture and an embossed emblem to the truck body, and corrugations to the front and rear mudguards.

Performance Reconstruction timings for all edited objects in the paper are given in Table 1, using a single CPU core on an i7-4790k@4.00GHz. The reconstruction of the diffuse environment is negligible compared to that of the reflection environment. Our partial reconstruction could be easily optimized with a parallel implementation. The performance of the hole-filling process highly depends on the size of the hole; it could be greatly improved by using an adaptive sphere tesselation strategy. Nevertheless, reconstruction is not too performance-demanding as it is done only once in pre-process. Most importantly, recompositing works in real-time in all our examples.

6. Discussion and future work

We have demonstrated a first solution for the edition of surface shape at the compositing stage, based on environment reconstruction and real-time recompositing. Our solution is tailored to opaque objects. We have demonstrated it on homogeneous materials, but it would be easily extended to objects with spatially varying reflectances, provided that diffuse (resp. reflection) illumination and reflectance buffers are output *separately* by the renderer. Our method would then be used to reconstruct the diffuse (resp. reflection) illumination buffers, and shading would be obtained through multiplication by reflectance at the recompositing stage.

However, our method would not readily work with transparent or translucent objects, since the complex light paths inside object volumes would forbid the reconstruction of prefiltered environment maps. Our approach also explicitly discards inter-reflections on a same object using occlusion buffers, but deals with inter-reflections among different objects as long as they are not too much extended in the scene. For instance, if the rear and front mudguards of Figure 1 where connected together, a single pre-filtered shading environment would not be enough. To deal with this issue, we would like to explore reconstructing multiple shading environments for extended objects and recombining them through warping.



Original/modified normals | Before/after shape editing | With edited material

Figure 11: *The left column shows the original normal buffer of the* `Table top` *scene (top), as well as a modified version (bottom) where a noise texture, an embossed symbol and a carved pattern have been respectively applied to the cup, kettle body and vase. In the middle column, we show the result of our approach (bottom) on the corresponding color image (top), using the reconstructed diffuse and reflection environments shown in Figure 12 (*1*st row). In the right column, we have edited the intensities of the diffuse and reflection components in both the original and modified scene. Reflections of nearby objects become more clearly apparent, as is also seen in Figure 12 (*2*nd row).*

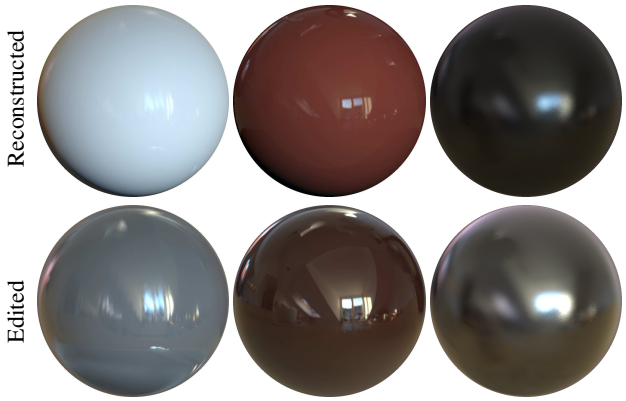


Figure 12: *Combined diffuse and reflection environments reconstructed from the cup, kettle and vase objects of Figure 11. The bottom row shows edited materials where the diffuse term is divided by* 4 *and the reflection term multiplied by* 4*: the supporting table and other nearby objects appear more clearly in the reflections.*

Reconstructed env. | After hole-filling | After regularization

Figure 13: *The reconstructed reflection environment for the spoon object in Figure 11 does not contain enough shading information to grant editing, even after hole-filling and regularization.*

Reconstruction is also restricted by object curvatures. If object shape is too simple, it will not provide enough shading information, which will require to fill in wider regions. This is illustrated in Figure 13, which shows the reconstructed reflection environment for the spoon in the `Table top` scene of Figure 11. In contrast, if object shape is too detailed with respect to image resolution, it will tend to reduce the accuracy of the reconstructed shading, as seen in the right column of Figure 8 when object complexity is increased. Nevertheless, as demonstrated in Section 5, our approach gives satisfying results in practical cases of interest, granting interactive exploration of local shape variations with real-time feedback. We believe it could already be useful to greatly shorten trial and error decisions in product design and movie production. A natural next step will be to manipulate surface positions instead of normals, in effect modifying shading buffers as if displacement maps were applied to objects. Our current approach does not permit to reproduce the associated visibility effects, such as cast shadows, local inter-reflections and modified silhouettes.

In this paper, we have considered out-of-the-box render buffers and have focused on algorithms working at the compositing stage. In future work, we would also like to work at the rendering stage to export more detailed shading information while retaining a negligible impact on rendering performance.

Acknowledgements

This work was supported by the EU Marie Curie Initial Training Network PRISM (FP7 - PEOPLE-2012-ITN, Grant Agreement: 316746) to C.J. Zubiaga and P. Barla. The authors would like to thank The Foundry for providing MODO licenses and assets (objects and environment maps).

References

[Bir05] BIRN J.: *Digital Lighting and Rendering (2Nd Edition)*. New Riders Publishing, Thousand Oaks, CA, USA, 2005. 1

[Bli78] BLINN J. F.: Simulation of wrinkled surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1978), SIGGRAPH '78, ACM, pp. 286–292. 2

[GI83] GOLDFARB D., IDNANI A.: A numerically stable dual method for solving strictly convex quadratic programs. *Math. Programming* (1983), 1–33. 3

[HS98] HEIDRICH W., SEIDEL H.-P.: View-independent Environment Maps. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware* (1998), Spencer S. N., (Ed.), The Eurographics Association. 2, 3

[IGG*14] IHRKE I., GRANIER X., GUENNEBAUD G., JACQUES L., GOLDLUECKE B.: An Introduction to Optimization Techniques in Computer Graphics. Eurographics 2014 - Tutorials, 2014. 4

[Joh02] JOHNSTON S. F.: Lumo: Illumination for cel animation. In *Proceedings of the 2Nd International Symposium on Non-photorealistic Animation and Rendering* (New York, NY, USA, 2002), NPAR '02, ACM, pp. 45–ff. 7

[KRFB06] KHAN E. A., REINHARD E., FLEMING R. W., BÜLTHOFF H. H.: Image-based material editing. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), SIGGRAPH '06, ACM, pp. 654–663. 2

[LBS06] LANGER T., BELYAEV A., SEIDEL H.-P.: Spherical barycentric coordinates. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 81–88. 4

[LN12] LOMBARDI S., NISHINO K.: Reflectance and natural illumination from a single image. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI* (Berlin, Heidelberg, 2012), ECCV'12, Springer-Verlag, pp. 582–595. 2

[LRR*14] LOCHMANN G., REINERT B., RITSCHEL T., MÃIJLLER S., SEIDEL H.-P.: Real-time Reflective and Refractive Novel-view Synthesis. In *VMV 2014: Vision, Modeling and Visualization* (Darmstadt, Germany, 2014), Bender J., Kuijper A., von Landesberger T., Theisel H., Urban P., (Eds.), Eurographics Association, pp. 9–16. 2

[RFWB07] RAMANARAYANAN G., FERWERDA J., WALTER B., BALA K.: Visual equivalence: Towards a new standard for image fidelity. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[RH01a] RAMAMOORTHI R., HANRAHAN P.: An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 497–500. 3

[RH01b] RAMAMOORTHI R., HANRAHAN P.: A signal-processing framework for inverse rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2001), SIGGRAPH '01, ACM, pp. 117–128. 2

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7 (2013), 121–130. 2

[ROTS09] RITSCHEL T., OKABE M., THORMÄHLEN T., SEIDEL H.-P.: Interactive reflection editing. In *ACM SIGGRAPH Asia 2009 Papers* (New York, NY, USA, 2009), SIGGRAPH Asia '09, ACM, pp. 129:1–129:7. 2

[SMGG01] SLOAN P.-P. J., MARTIN W., GOOCH A., GOOCH B.: The lit sphere: A model for capturing npr shading from art. In *Proceedings of Graphics Interface 2001* (Toronto, Ont., Canada, Canada, 2001), GI '01, Canadian Information Processing Society, pp. 143–150. 2, 5

[Tok15] TOKUYOSHI Y.: Specular lobe-aware filtering and upsampling for interactive indirect illumination. *Comput. Graph. Forum 34*, 6 (2015), 135–147. 2

[VB15] VERGNE R., BARLA P.: Designing Gratin, A GPU-Tailored Node-Based System. *Journal of Computer Graphics Techniques 4*, 4 (2015), 17. 7

[VBFG12] VERGNE R., BARLA P., FLEMING R. W., GRANIER X.: Surface flows for image-based shading design. *ACM Trans. Graph. 31*, 4 (July 2012), 94:1–94:9. 2

[ZMB*15] ZUBIAGA C. J., MUÑOZ A., BELCOUR L., BOSCH C., BARLA P.: MatCap Decomposition for Dynamic Appearance Manipulation. In *Eurographics Symposium on Rendering 2015* (Darmstadt, Germany, June 2015). 2

[ZRJ*15] ZIMMER H., ROUSSELLE F., JAKOB W., WANG O., ADLER D., JAROSZ W., SORKINE-HORNUNG O., SORKINE-HORNUNG A.: Path-space motion estimation and decomposition for robust animation filtering. *Computer Graphics Forum 34*, 4 (2015), 131–142. 2

Appendix

For completeness, we present from left to right the surface IDs, ambient and specular occlusion buffers, first for the `Table top` scene, then for the `Truck` scene.