Vis Comput (2017) 33:129–138
DOI 10.1007/s00371-015-1163-2

ORIGINAL ARTICLE

RayPortals: a light transport editing framework

Thomas Subileau[1,2,3] · Nicolas Mellado[1,2,3] · David Vanderhaeghe[1,2,3] · Mathias Paulin[1,2,3]

Published online: 30 October 2015

Abstract Physically based rendering, using path-space formulation of global illumination, has become a standard technique for high-quality computer-generated imagery. Nonetheless, being able to control and edit the resulting picture so that it corresponds to the artist vision is still a tedious trial-and-error process. We show how the manipulation of light transport translates into the path-space integral formulation of the rendering equation. We introduce portals as a path-space manipulation tool to edit and control renderings and show how our editing tool unifies and extends previous work on lighting editing. Portals allow the artist to precisely control the final aspect of the image without modifying neither scene geometry nor lighting setup. According to the setup of two geometric handles and a simple path selection filter, portals capture specific lightpaths and teleport them through 3D space. We implement portals in major path-based algorithms (Photon Mapping, Progressive Photon Mapping and Bi-directional Path Tracing) and demonstrate the wide range of control this technique allows on various lighting effects, from low-frequency color bleeding to high-frequency caustics as well as view-dependent reflections.

Keywords Rendering · Global illumination · Editing · Manipulation · Physically based

✉ Thomas Subileau
subileau@irit.fr

1 Université de Toulouse, Toulouse, France

2 UPS, Toulouse, France

3 IRIT, Toulouse, France

1 Introduction

High-quality digital contents, such as movies, rely on the ability and creativity of the artist to fully exploit the capabilities offered by content creation and rendering software. In the current content creation workflow, setups of the scene parameters (e.g. geometry, material, camera) and light parameters are separated. Once the scene parameters are entirely fixed, lighting designers add and tune light sources one by one [2].

Even though lighting parameters are perfectly set, the physically based rendering does not necessarily match the artistic goal. In other words, the artistic freedom is limited to the physical simulation of light transport. As a consequence, methods have been developed to either modify the renderer input (scene configuration) or output (layered images) to obtain a desired result.

Even if widely used, post-processing techniques are limited to image-based processing and cannot provide the full set of changes the artists would need. On the other hand, the lighting design is also tweaked and tricked, adding for instance lights not casting shadows [4], only giving specular lighting or lights that only affect a given object (i.e. light linking). The involved trial-and-error process is tedious, requires a lot of experience and several time-consuming re-computations of physically based renderings.

Following the pioneer methods exploring the editing and control of direct illumination, recent results propose to modify the global illumination of a scene in order to add artistic control through the process. However, despite their efficiency, these approaches are limited to a restrained set of effects.

In order to tackle these limitations, we propose to edit lighting effects directly in *path-space*, i.e. by changing how light propagates in the scene. These changes affect only the rendering stage and do not require to edit either the geometry or the lighting setup.

Our key idea is to allow artists to directly manipulate the light propagation (e.g. geometrical optics) of user-selected lightpaths through the use of *portals* that capture and *teleport* lightpaths through 3D-space (Figs. 1, 2). We define portals as a manipulator composed of an input surface, a selection filter and an output surface. Every path that hits the input surface and corresponds to the user-defined path filter is teleported to the output surface.

While modifications might not be physically correct, we still rely on physically based renderers and edits thus remain visually coherent in the resulting rendering. Also, if not intentionally modified by the user, the light energy is conserved and sampling functions are not modified. This ensures the conservation of consistence and convergence properties of the original rendering technique.

We present in this paper two main *contributions*: a reformulation of the path-space integral enabling light propagation editing (Sect. 3.2), and a mechanism to alter light transport which we call *portals* (Sect. 3.3). Our formulation is versatile and allows a wide range of manipulations. For instance, we show that existing related techniques can be defined as specific portal configurations. By definition, portals are totally decorrelated from the scene description. They can be easily tuned and animated with keyframing. The technique correctly handles shadow rays (Sect. 4) and could be integrated in any path-based renderer.

Finally, we show how our approach can be used for advanced modifications of complex lighting effects resulting from global illumination (Sect. 5).

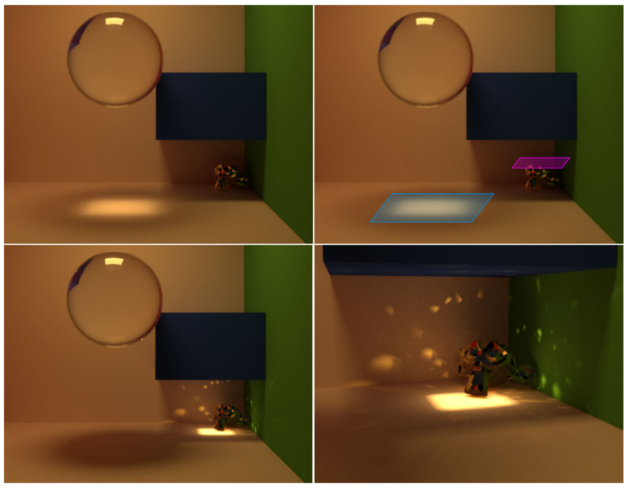


Fig. 1 Portals allow the manipulation of lighting effects in a scene. Here, the caustic created by the sphere is captured (*blue portal*) and moved (*pink portal*) under the cube, revealing the shiny Suzanne monkey. The edit is smoothly integrated in the rendering of the scene

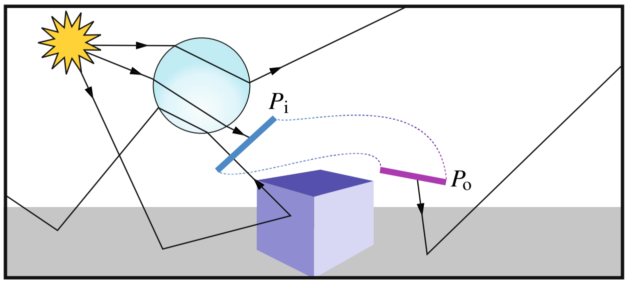


Fig. 2 Portals alter the light propagation. When a ray intersects P_i (*in blue*) and matches the selection filter, it is *teleported* to P_o (*in pink*)

2 Previous work

In the context of digital content creation, artists have to manually setup materials and lights parameters of a 3D scene to obtain at the end the expected rendering effects. This work can be tedious and requires to estimate correctly how light interacts and propagates in the scene.

Previous work on intuitive editing allows user to paint an expected lighting effect in the scene, and optimizes its configuration to produce an as-correct-as-possible rendering output [15–17, 19]. These approaches provide efficient design interfaces, but remain limited to the impact of scene parameters on the final rendering. This impact can be hard to control, especially in a global illumination context. Our work is orthogonal to these methods as we choose not to modify the lights and materials description, but only edit light propagation during the rendering process. Several editing methods [12, 14, 18] focus on shadow manipulation. These methods provide different means to control shadows shape, position and smoothness. We focus on the editing of light transport, shadows shape cannot be directly manipulated in our approach.

Seminal works in lighting editing algorithms, limited to either direct lighting or specular reflection have inspired our work. Extending early work of Barzel [1], BendyLights [9] offers a direct lighting manipulation by bending the light propagation for spotlights and hence facilitates the control of direct illumination effects. Ritschel et al. [21] allow the user to change the reflected direction of purely specular surfaces.

Ritschel et al. [22] define on-surface signal deformation to manipulate the appearance of any signal over the surface of 3D objects. The manipulation itself stays on surface and is prone to sliding artifacts when applied on animated surfaces. As our approach is totally defined in 3D-space, we do not have these kinds of drawbacks. Using a robust visualization tool, Reiner et al. [20] show that light propagation can be well comprehended and particle flows creating specific lighting effects can be spatially and semantically clustered. Following this approach, Schmidt et al. [23] propose a path retargeting technique: the user manipulation of a *shading effect* (referred as lighting effect in our paper) affects the outgoing tangent frame of the previous interaction surface. While allowing various edits, there are some limitations in the freedom of the manipulation. For instance, moving a lighting effect below or through an object is not achievable as the object will intersect the edited paths. To overcome this limitation, the authors chose to add a specific proxy-object approach to specify that an object does not interact with the edited paths.

We define a general theoretical framework for path-space editing. We also formulate previous approaches [22,23] in this framework. We propose a manipulator which provides a flexible mechanism to edit path space. This mechanism allows artists to position the manipulator anywhere between the source surface (as in Schmidt et al. [23]) and the destination surface (as in Ritschel et al. [22]) of an effect and handles all sort of situations inbetween. This editing freedom tackles the previous works' limitations stated above.

3 Editing light propagation in path-space

3.1 Path-space formulation

As introduced by Veach [24], the light transport problem can be expressed as an integral over path-space. This path-integral defines the color of a pixel on the computed image as the integral of the flux transported by each lightpath from sources to this pixel. The measurement I_j for each pixel j of an image is written as

$$I_j = \int_{\Omega} f_j(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{1}$$

with Ω the set of all transport paths also called *path-space*, and $\mathrm{d}\mu$ the area-product measure of a path $\bar{\mathbf{x}}$. Paths are of the form $\bar{\mathbf{x}} = x_0 x_1 \ldots x_k$, with $1 < k < \infty$ and $x_i \in \mathscr{S}$, $\mathscr{S}$ being the union of all surfaces. The ith segment $\langle x_i, x_{i+1} \rangle$ of a path is a straight line between the two consecutive positions x_i and x_{i+1}. The contribution of a lightpath is measured as

$$\begin{aligned} f_j(\bar{\mathbf{x}}) = {} & L(x_0 \to x_1) G(x_0 \leftrightarrow x_1) \\ & \times \left(\prod_{i=1}^{k-1} f_s(x_{i-1} \to x_i \to x_{i+1}) G(x_i \leftrightarrow x_{i+1}) \right) \\ & \times W(x_{k-1} \to x_k), \end{aligned}$$

with L the emitted light, G the geometric or propagation term, f_s the scattering function (e.g. the *bsdf*) and W the importance of the path for the pixel j.

If we ignore participating media, the propagation of the light between two surfaces along a path segment is influenced only by the relative incoming and outgoing light directions, the length of the segment, and the surfaces visibility V at the interaction points:

$$G(x_i \leftrightarrow x_j) = V(x_i \leftrightarrow x_j) \frac{(n_i \cdot |x_j - x_i|)\,(n_j \cdot |x_i - x_j|)}{||x_i - x_j||^2},$$

with $\cdot$ the dot product, $|x| = \frac{x}{||x||}$, n_i the normal vector of the surface at x_i, and $V(x_i \leftrightarrow x_j)$ the visibility function.

3.2 Path-based propagation editing

As seen in Sect. 2, lighting can be edited by changing the light properties L, the scattering functions f_s or the propagation G. Artists usually tune lights and materials properties to obtain a desired lighting effect while propagation is left untouched. In this work, we propose a new formalism enabling the definition of a large panel of transformations to modify how light propagates in the scene, independently of the lighting and material configuration. The general idea is to capture light flux somewhere in the scene and release it somewhere else, this idea is translated in the path-integral as a modification of the propagation function.

Changing how light is transmitted without changing materials and lights is equivalent to move a light contribution from a point x_j to another point $x_e \in \mathscr{S}$. This can be done by modifying the propagation function $G(x_i \leftrightarrow x_j)$ to a new editable propagation function $\overline{G}(x_i \leftrightarrow x_j, x_e)$, where the contribution reaching x_j is moved to x_e. As shown in Fig. 3, modifications of the light propagation can be applied from any point $p_e = x_i + t|x_j - x_i|$ on the segment $\langle x_i, x_j \rangle$.

Let now assume that we know the edited receiver position x_e and the editing position p_e. To edit the propagation of the initial lightpath, we remove its contribution at the position x_j and add it at the new receiver position x_e. As a consequence, the propagation between x_i and x_j is cancelled, hence $G(x_i \leftrightarrow x_j) = 0$. The light is now transmitted to x_e and combined to other incoming paths. We now present how this concept translates to the lightpath contribution measure $f_j(\bar{\mathbf{x}})$, and then focus on the definition of $\overline{G}$.

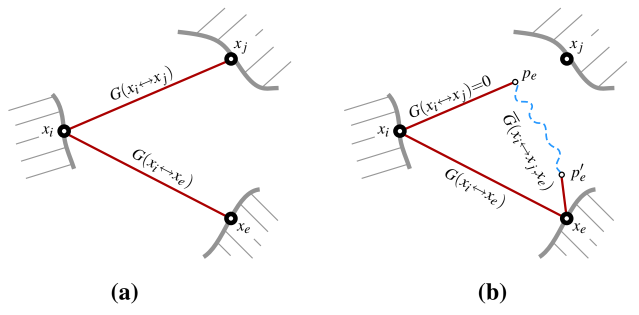

x_j
$G(x_i \leftrightarrow x_j)$
x_i
$G(x_i \leftrightarrow x_e)$
x_e
(a)
x_j
p_e
$G(x_i \leftrightarrow x_j)=0$
$\overline{G}(x_i \leftrightarrow x_j, x_e)$
x_i
p'_e
$G(x_i \leftrightarrow x_e)$
x_e
(b)

Fig. 3 **a** Non-edited scene, where the light propagation along segments $\langle x_i, x_j \rangle$ and $\langle x_i, x_e \rangle$ is computed using G. **b** Edited scene: propagation along $\langle x_i, x_j \rangle$ is cancelled at p_e, and transformed to p'_e in order to finally fall on x_e. This whole process is defined by the edited propagation term $\overline{G}$. *Red lines* represent propagation segments, potentially shared between multiple lightpaths

Edited ligthpath propagation We want to modify only how the light is received, so the scattering function f_s is evaluated at x_i with the initial incoming and outgoing directions, even for edited segments. Thus, in the path contribution measurement of a lightpath (see Sect. 3.1), we replace the term $f_s(x_{i-1} \to x_i \to x_{i+1})G(x_i \leftrightarrow x_{i+1})$ with

$$f_s(x_{i-1} \to x_i \to x_{i+1})G(x_i \leftrightarrow x_{i+1}) \\ + \sum_{n=1}^{m_i} f_s(x_{i-1} \to x_i \to x^n_{i\leftrightarrow i+1})\overline{G}(x_i \leftrightarrow x^n_{i\leftrightarrow i+1}, x_{i+1}),$$

where m_i is the number of segments starting from x_i and edited to x_{i+1}. $\overline{G}$ is the editing propagation function describing how the light propagation from x_i to the nth original unedited position $x^n_{i\leftrightarrow i+1}$ is modified to reach x_{i+1}, as depicted by the dashed blue line in Fig. 3. We apply an analogous modification to the first segment $x_0 \leftrightarrow x_1$ and note $\overline{f}_j(\bar{\mathbf{x}})$ the resulting contribution measure (see full definition in "Appendix").

Editable transmission function One could define $\overline{G}(x_i \leftrightarrow x_j, x_e)$ as $G(x_i, x_e)$, but as shown in Fig. 4a, this covers only a subset of the possible modifications. As already stated, we want to modify only how the light is received, so we need to keep $|x_j - x_i|$ as outgoing direction to evaluate the propagation from x_i. We note n_e the normal vector at x_e, and define the editable propagation function as

$$\overline{G}(x_i \leftrightarrow x_j, x_e) = V(x_i \leftrightarrow p_e)V(p'_e \leftrightarrow x_e) \\ \times \frac{(n_i \cdot |x_j - x_i|)\ (n_e \cdot |d_e|)}{(||x_i - p_e|| + ||p'_e - x_e||)^2}, \qquad (2)$$

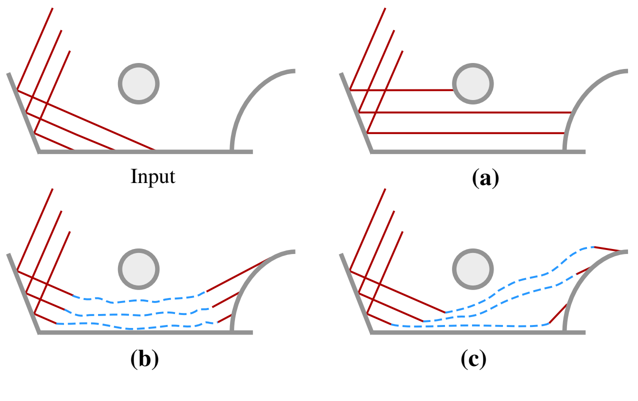


Fig. 4 Several path-space modifications configuration. An input scene is modified by changing how light is transmitted to a target surface. *Red segments* are actual lightpaths. *Blue dashed lines* correspond to lightpath transformations. By increasing complexity: **a** light rays are re-oriented. **b** Light field is sliced and *teleported*. **c** Each ray is *teleported* and locally re-oriented according to the geometry

where d_e is the edited incoming light direction at x_e. We define p'_e and d_e as

$$p'_e = \mathscr{M}_{x_j,x_e}(p_e) \\ d_e = \mathscr{M}_{x_j,x_e}(p_e) - x_e, \qquad (3)$$

where $\mathscr{M}_{x_j,x_e}$ transforms the light segment initially coming in x_j to x_e. $\mathscr{M}_{x_j,x_e}$ is what we call a *portal*, and can be configured to obtain a wide range of modifications of the light propagation, described in the next section. By construction, the non-edited scenario can be retrieved by setting $\mathscr{M}_{x_j,x_e}$ as the identity function and x_e as x_j.

3.3 Portal-based propagation editing

According to Reiner et al. [20], humans are good at identifying and grouping lightpaths corresponding to lighting effects (e.g. caustics) in the 3D scene. Assuming a path selection mechanism, portals can be applied in two ways on groups of lightpaths.

In a general setting, one can define $\mathscr{M}_{x_j,x_e}$ *per-path* in order to locally adapt the transformation to the scene, e.g. the geometry surrounding x_j and x_e. Here the artistic freedom is maximal, however care must be taken to define a practicable transformation according to path-based rendering algorithms. For instance, Ritschel et al. [22] evaluate lighting at x_e as if it is physically located at x_j (see Fig. 4c). Hence $p_e = x_j$ and d_e has the same angle with n_e than $|x_i - x_j|$ with n_j. As a result, $\mathscr{M}_{x_j,x_e}$ defines a mapping of the shading configuration from x_j to x_e. This function is not trivial to define, and is computed by optimization in Ritschel's approach.

In a simplified setting, the propagation of the light can be edited uniformly within a group of paths, in that case we note $\mathscr{M}$ the functional transformation applied to a group of paths. According to Reiner et al. [20], grouped paths usually share common properties (e.g. geometry), and can thus be edited uniformly to transform the resulting lighting effect. For instance, Schmidt et al. [23] propose to rotate the output direction from x_i to reach x_e (see Fig. 4a). In other words, $\mathscr{M}$ is defined as the identity function and $p_e = x_i$, thus

$$p'_e = \mathscr{M}(p_e) = x_i \\ d_e = \mathscr{M}(p_e) - x_e = x_i - x_e. \qquad (4)$$

Using our formalism one can see that both aforementioned existing techniques represent very specific edits and do not span a wide range of modifications of the light propagation.

We propose to edit the light propagation with a more expressive portal definition, according to the following constraints. First, portals must be compatible with any path-based rendering algorithm, so we need to ensure that $\mathscr{M}_{x_j,x_e}$

is invertible. This allows bi-directional path traversal, a step usually required to compute shadow rays. Second, we want to include the path selection mechanism in our definition, to provide a unified path-based selection and editing metaphor. Third, we want to combine both geometric and photometric effects at once.

Geometrically, we define portals as a pair of surfaces parametrized over the same domain, the former P_i grabbing incoming light segments and the latter P_o releasing them somewhere else in the scene. Selecting a group of paths is achieved by intersecting them with P_i. The resulting intersection points define the editing positions p_e used in Eq. 2. We note $\mathscr{G}_{x_j,x_e}$ the invertible geometric transformation moving a segment from x_j to x_e through P_i and P_o. We also define $\mathscr{R}_{x_j,x_e}$ an arbitrary photometric transformation of the incoming light segment, leading to

$$\mathscr{M}_{x_j,x_e}(p_e) = \mathscr{G}_{x_j,x_e}(p_e)\mathscr{R}_{x_j,x_e}(p).$$

In practice, the complexity of $\mathscr{G}_{x_j,x_e}$ can be adapted according to the surfaces geometric properties. For instance, uniformly parameterized planar surfaces define $\mathscr{G}_{x_j,x_e}$ as a linear transformation matrix encoding the rotation, translation and scale between multiple surfaces, noted M in the following. We can attach to portals additional embedded functions to easily supplement the definition of $\mathscr{M}_{x_j,x_e}$. These transformations may be used to modify both the geometric term $\mathscr{G}_{x_j,x_e}$ and the photometric term $\mathscr{R}_{x_j,x_e}$ (Sect. 5).

Finally, spatial path selection can be extended with a semantic path selection using *path regular expressions*. This notion has been introduced by Heckbert [6] and extended by Veach [24]. Each path is characterized by a set of symbols, each symbol represents the interaction that occurs at a vertex of the path. Regular expressions based on these symbols can then be attached to any portal and used to filter independently each incoming paths. We use the syntax proposed by Schmidt et al. [23] that adds light and object identifiers to each interaction, plus a specific symbol to represent portal traversal. Since paths sharing similar scattering events and similar trajectory in 3D-space produce a coherent variation of shading in the 3D scene, we formally define a lighting effect by the set of lightpaths going through a given region in 3D-space, i.e. intersecting P_i, and matching a given regular expression.

4 Implementation

We have implemented portals in Mitsuba software [7] for three rendering algorithms: photon mapping [8], Progressive photon mapping [5] and Bi-directional Path Tracing [10]. We have interfaced portals with Blender [3] and Mitsuba export addon to allow an interactive setup and manipulation of portals. For each portal, we add a manipulator in the 3D scene defined as a pair of planar surfaces P_i, P_o parameterized over the same domain, typically $[0, 1]^2$. We note $P_\mathrm{i}(u, v)$ a point defined by the parametric coordinates (u, v) on P_i.

Theoretically, the path-integral formulation of light transport considers the path-space Ω to be entirely known. In practice, it is partially evaluated by sampling. The implemented algorithms construct samples of Ω with a two-step routine:

- During the first step, paths are built by casting rays from a source (light or camera) toward the scene and bouncing iteratively on the geometry. The correct evaluation of $\overline{G}$ during this step is direct. When a ray intersects P_i at position $p_e = P_\mathrm{i}(u, v)$, it is recast from edited position p'_e in the same direction relatively to the local tangent frame.
- During the second step, a shadow ray is cast between each couple of vertices x_1 and x_2 to evaluate if the segment $\langle x_1, x_2\rangle$ transports energy. In Photon mapping, this corresponds to final gathering whereas in Bi-directional Path Tracing, this corresponds to the connection step between the camera subpath and the light subpath.

When using portals, we need to correctly evaluate the propagation between x_1 and x_2. As shown in Fig. 3, we need to evaluate $G(x_1 \leftrightarrow x_2)$ and $\overline{G}(x_1 \leftrightarrow x_j, x_2)$ for every edits. Evaluating G is done as usual by casting a ray from x_1 toward x_2. Evaluating $\overline{G}$ corresponds to finding rays that leave x_1 toward unknown points x_j and intersect P_i such that, when leaving P_o, they land on x_2. Figure 5 represents the geometric setup of this case. Finding all the shadow rays that connect the two vertices ensures the correct evaluation of the rendering equation.

To solve this problem, we define $g(u, v) \to \mathbb{R}^+$ a function that returns the distance between a position along the ray $r(u, v)$ and the vertex x_2, with $r(u, v) = P_\mathrm{o}(u, v) +$

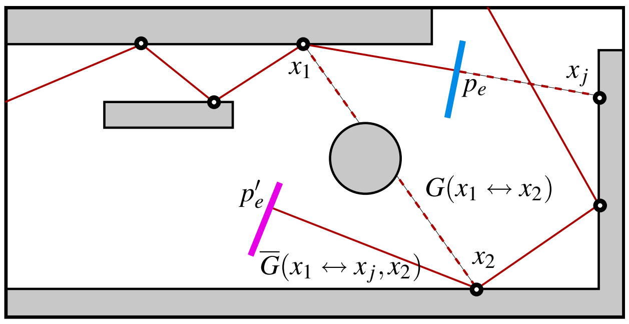

x_1
p_e
x_j
$G(x_1 \leftrightarrow x_2)$
p'_e
$\overline{G}(x_1 \leftrightarrow x_j, x_2)$
x_2

Fig. 5 When evaluating the propagation between two points, we need to take portals into account. Here, $G(x_1 \leftrightarrow x_2)$ is nil but going through the portals, a shadow ray lands on x_2 and transmits light from x_1 to x_2

$t\,M(P_\mathrm{i}(u,v)-x_1)$ and $t \in \mathbb{R}^+$. The set of rays possibly connecting x_1 and x_2 are defined by $(u,v) \in \ker(g)$, i.e. (u,v) such that $g(u,v)=0$.

We have implemented the solution for our planar portal objects. Considering the points p_e and p'_e defined as:

$$p_e = P_\mathrm{i}(u,v) = O_\mathrm{i} + u\,\mathbf{u_i} + v\,\mathbf{v_i}$$
$$p'_e = P_\mathrm{o}(u,v) = O_\mathrm{o} + u\,\mathbf{u_o} + v\,\mathbf{v_o}$$

with O_i the origin of portal P_i and $\mathbf{u_i}$, $\mathbf{v_i}$ the parameterization vectors, respectively, O_o, $\mathbf{u_o}$ and $\mathbf{v_o}$ for portal P_o, we are searching for u, v and t such that:

$$x_2 = p'_e + t M(p_e - x_1)$$

Multiplying both sides by M^{-1}:

$$M^{-1}x_2 = p_e + t(p_e - x_1)$$

Developing and factorizing by p_e:

$$\frac{1}{1+t}M^{-1}x_2 + \frac{t}{1+t}x_1 = p_e.$$

This corresponds to the intersection between the segment $\langle x_1, M^{-1}x_2\rangle$ and the surface of P_i, as shown in Fig. 6. We solve this equation using the algorithm presented by Lagae and Dutré [11]. After finding the (u,v) coordinates, we need to evaluate $V(x_1 \leftrightarrow p_e)$ and $V(p'_e \leftrightarrow x_2)$ to verify that no geometry blocks the visibility.

5 Results

In this section, we will present several results showing how portals can reproduce previous work (Sect. 5.1) as well as other various geometric (Sect. 5.2) and photometric (Sect. 5.3) transformations. In any example, portals are freely positioned in 3D-space and are thus highly independent from the scene. This is shown in Fig. 7 where the same portal is applied to different geometries.

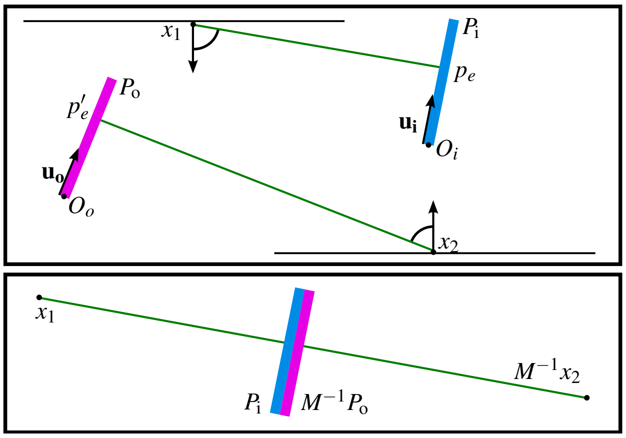


Fig. 6 We are searching the ray leaving x_1, intersecting P_i and transformed by the portal such that, when leaving P_o, it lands on x_2. It is defined by the intersection between the segment $\langle x_1, M^{-1}x_2\rangle$ and P_i

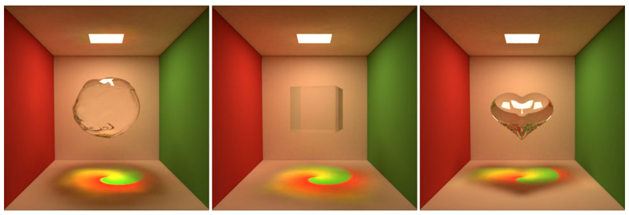


Fig. 7 The same portal is used in a scene with three different objects projecting a caustic. The modification applies consistently and shows that portals are highly independent from the scene

5.1 Implementation of previous work

Ritschel et al. [21] allow the user to define what part of the scene is seen through mirror reflections. This translates into editing the propagation term $G(x_{k-1} \leftrightarrow x_k)$ in a path of length k and only if the interaction at x_{k-1} is a specular reflection. Such edits are handled with portals filtering ES paths. Examples are shown in the supplemental video.

As shown in Sect. 3, Schmidt et al. [23] translates smoothly in our formulation. We show an example of light retargeting using portals in Fig. 8b. Here, M is defined as the identity matrix and the edited rays are rotated using a constant function embedded in the portal. Light retargeting does not conserve the geometric terms in the computation of $\overline{G}$, i.e. either the distance or the relative directions to the surfaces are preserved (see Eq. 2). Therefore, after retargeting, the illuminance of the lighting effect is not conserved and the resulting caustic is visually different from the original. In Fig. 8c, we move the output portal in order to conserve all terms for one point x_i. For other points, terms will vary depending on the curvature of the surfaces. We can see that after *teleportation*, the caustic illuminance remains visually similar to the original.

5.2 Geometric transformation

A typical editing of the propagation using portals is illustrated in Fig. 1. Caustic paths are *teleported* onto the Suzanne model. We can see that, after transformation, paths continue to interact with the geometry, hence creating caustic sparkles on the walls due to the facetted model. As we evaluate the propagation toward the original position, the blue cube does not block the propagation. The setup of this portal is shown in the supplemental video.

Portals also allow to replicate a lighting effect. To do this, we associate in a portal one input surface to multiple out-

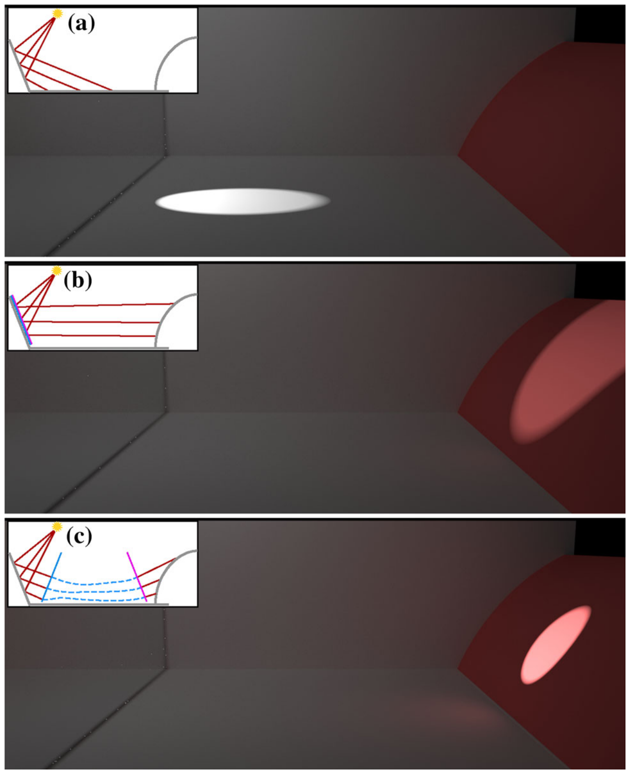


Fig. 8 Application of Fig. 4. **a** Original rendering. **b** Light retargeting with portals. **c** Light field *teleportation* with portals

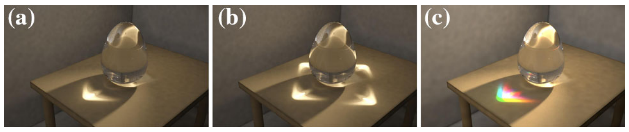


Fig. 9 A portal captures the caustic and replicates it with multiple output surfaces, disposed around the glass egg. **a** Original scene. **b** Duplication. **c** Duplication and shifting spectrum hue

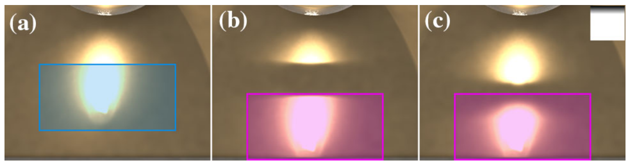


Fig. 10 Using a stochastic map allows to progressively select and transform the paths. **a** Original result. **b** Raw editing. **c** Smooth editing

put surfaces, each resending captured rays. Figure 9b shows how a lighting effect is copy/pasted in the scene. Duplicated output surfaces act as new lights and thus do not conserve overall energy.

Portals are finite surfaces and may create sharp discontinuities when overlapping lighting effect boundaries. For instance, if the input portal covers only one part of the caustic (Fig. 10a), it will create a sharp visual discontinuity after editing, as shown in Fig. 10b. A soft selection can be done using a stochastic function defining the probability for a ray

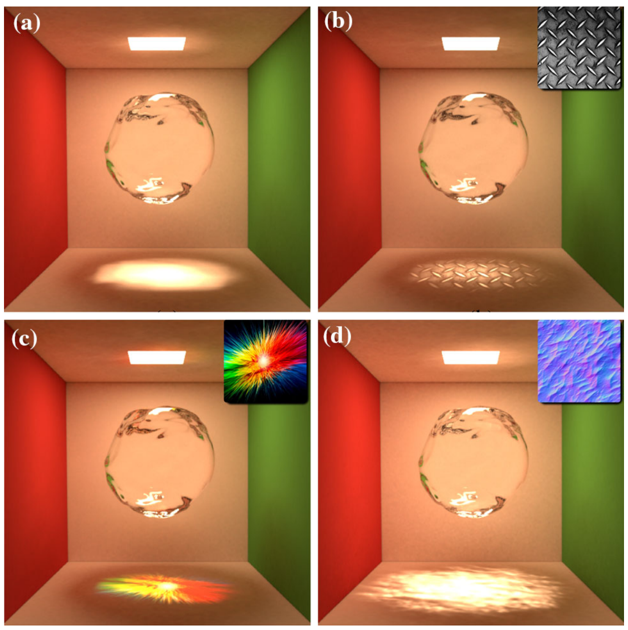


Fig. 11 Using textures to control lighting effect. **a** Unmodified scene. **b** Using a texture to control the intensity. **c** Using a texture to control the hue. **d** Using a normal map to modify outgoing directions

to be edited or not. In practice, this function is defined using a grayscale texture. Figure 10c shows how such a probability map allows to smoothly apply the editing when overlapping a lighting effect.

A normal map can also be used to modify the outgoing local tangent frame and thus tilt rays directions as shown in Fig. 11d. In this case, M is modified ununiformly and is not a linear transformation anymore, hence shadow rays are not edited by portals in this result.

5.3 Photometric transformation

Textures can also be used to transform the intensity (Fig. 11b) or the hue (Fig. 11c) of lightpaths. An example of duplication coupled with different photometric transformations is shown in Fig. 9c. The statue example (Fig. 12) shows how a color bleeding is colored. Portal input and output surfaces are placed in front of the statue head and a texture is used to transform the color bleeding hue from the green wall. Other examples of photometric transformations are shown in Fig. 7 and in the supplemental video.

5.4 Performance

In our experimental implementation, using portals adds two types of computational overhead:

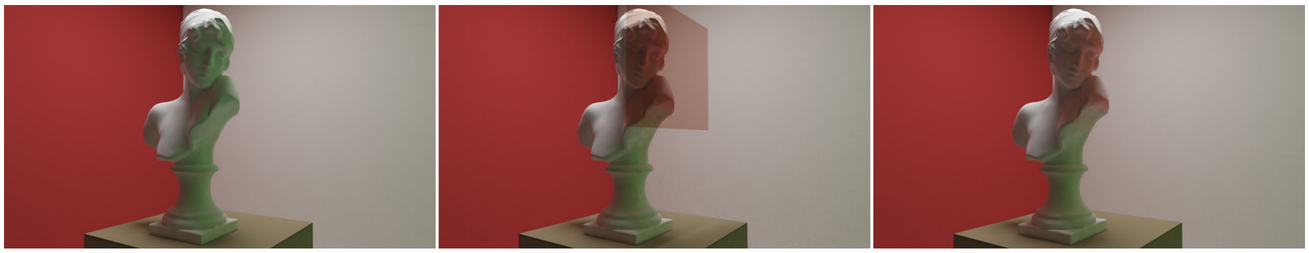


Fig. 12 Color bleeding is edited through a spectral transformation. The regular expression filter is LD. There is a color shift to transform the greenish bleeding to a reddish one. From *left* to *right* unmodified scene, portal surfaces (*in red*), result

Table 1 Performance table for Fig. 1 in Photon Mapping for two different quantities of caustic photons

	$\sim$ 270 M photons		$\sim$ 500 M photons	
	Original	Edited	Original	Edited
Rendering time (s)	264.61	319.46	431.22	516.00
Overhead (%)	–	20.73	–	19.66

Original is top left figure, edited is bottom left figure

Table 2 Performance table for Fig. 1 in bi-directional Path Tracing for two levels of samples per pixels (spp)

	1024 spp		8100 spp	
	Original	Edited	Original	Edited
Rendering time (m)	15.79	17.64	125.26	141.64
Overhead (%)	–	11.74	–	13.07

Original is top left figure, edited is bottom left figure

1. a selection cost depending on the number of rays intersecting the portals, and due to the regular expressions matching evaluation,
2. an editing cost depending only on the number of edited rays, caused by the modification of the rays during their interaction with the portal.

For the edited rendering shown in Fig. 1, the combination of the constant and varying overheads represents around 12 % of the non-edited timings, as shown in Table 2. It remains the same when using more samples (e.g. 1024 or 8100 samples per pixels), since both for the same scene rendered using Photon Mapping, the overhead is slightly more important and raise to 20 %, as shown in Table 1. We can see that for both techniques, the overhead is relatively constant for a given scene, and not impacted by the number of samples.

In a second experiment, we measured the variation of overhead according to the number of edited rays. For a fixed geometry and light setups, we applied a stochastic function on the portal in order to vary the amount of selected rays, as done in Fig. 10 for smooth caustic selection. Tables 3 and 4 report the cost of rays selection and editing, and the total overhead for probability of selection ranging from 100 to 6 %, respectively, for Bi-directional Path Tracing and Photon Mapping. The bottleneck of our technique remains in the evaluation of the regular expression. This is understandable as regular expression matching tests are knowingly computationally expensive and need to be evaluated for every rays intersecting the surface, in order to know whether or not the ray needs to be edited.

5.5 Discussion

User control In the current implementation, portals are manipulated directly through the use of two geometric handles, similarly to the positioning of objects in 3D modeling software (as shown using Blender in the supplemental video). The regular expression filter can be defined using presets (e.g. mirror reflections are $ES+$, purely specular caustics are $LS+$) or manually. A more intuitive automatic user-interface could be envisioned. First, the user selects an area on the 3D surface, for instance, with a sketch in image-space. Second, the portal is extracted from paths going through this area : the regular expression from the paths prevailing syntax and the surface from the paths footprint. Third, the drag-and-drop of the area in the scene defines the transformation $\mathscr{G}_{x_j,x_e}$ associated to the portal. Such process is similar to the interface proposed and validated by Schmidt et al. [23].

Prefixed regular expression When a ray hits an input portal, we compare the expression of the path from its starting point to the last known vertex to decide if the path is to be transformed. The regular expression thus represents the path interactions *before* intersecting the portal and not the full path within the scene. A back-tracking mechanism would allow to compute light scattering events occurring after the portal surface and decide to filter the path afterward based on its full definition. However, we found in our experiments that capturing lighting effects with a prefix regular expression is effectively done.

Bi-directional traversal Consequently, portals apply either on light subpaths (filter starting with L) or camera subpaths (filter starting with E). It is possible for subpaths to intersect opposite-type filtering input portals backwards and then

Table 3 Detailed execution time, with various selection setting (bi-directional Path Tracing)

BDPT (81 spp)	Unedited	Selecting 100 %	Selecting 85 %	Selecting 40 %	Selecting 6 %
Rendering time (s)	74.23	82.44	82.10	81.40	80.83
Regexp overhead (%)	–	8.81	8.61	6.96	6.47
Editing overhead (%)	–	1.95	1.75	0.79	0.12
Overhead (%)	–	11.06	10.60	9.66	8.89

Table 4 Detailed execution time, with various selection setting (Photon Mapping)

PM (∼ 70 M caustic photons—1 spp)	Unedited	Selecting 100 %	Selecting 85 %	Selecting 40 %	Selecting 6 %
Rendering time (s)	56.35	65.47	65.21	62.38	61.92
Regexp overhead (%)	–	12.90	12.51	9.32	7.24
Editing overhead (%)	–	1.89	1.65	0.83	0.19
Overhead (%)	–	16.19	15.74	10.70	9.89

be linked with opposite-type subpaths, potentially making the full path eligible to the portal transformation. To address this problem, segments that cross opposite-type input portals backwards are tagged. After the linking step, tagged segments are reevaluated both ways and, if they match the filter, visibility is recomputed as explained Sect. 4.

6 Conclusion and future work

We have analyzed the problem of editing light transport and have shown how it can be uniformly defined within the path-integral formulation of the rendering equation. According to this definition, we have proposed a method called *ray portals* that allows the capture and modification of lightpaths both in 3D-space and path-space.

In the current implementation, portals do not handle participating media. However, the proposed formulation could be adapted by ensuring that the editing function $\mathscr{M}_{x_j,x_e}$ is continuous and calculable for any point p'_e between p_e and x_e. Following an approach similar to light beams manipulation [13] could be envisioned to do such extension.

Another interesting direction for future work is extending portals to more various geometries. Especially, non-planar geometries would allow the definition of complex editing function $\mathscr{M}_{x_j,x_e}$ as the one used in Ritschel et al. [22], fitted to match the target surface. It would allow edits as shown in Fig. 4c that locally conserve illuminance of a lighting effect through editing. It would also ease the capture and manipulation of multidirectional effects, such as low-frequency ambient lighting. However, to fully integrate within a path-based rendering framework, portal transformations need to be inverted which can be challenging for complex $\mathscr{M}_{x_j,x_e}$.

Ray portals propose a unified and efficient solution which allows complex editing of light transport phenomena through the manipulation of simple geometric portals and user-defined controls. The proposed method complements and extends previous state-of-the-art techniques, especially as it is largely scene independent and ensures a correct evaluation of light propagation.

Acknowledgements We especially thank the Observatory of Systems Information Retrieval and Indexing of Multimedia contents (OSIRIM) platform of the Toulouse Institute of Computer Science Research (IRIT).

7 Appendix

The edited path contribution measure is defined as

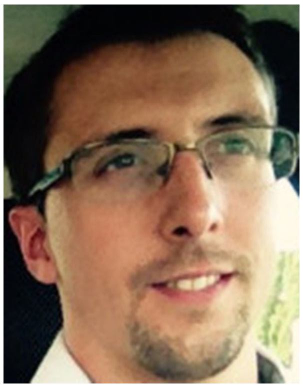
$$\overline{f}_j(\bar{\mathbf{x}}) = \left(\begin{array}{l} L(x_0 \to x_1)G(x_0 \leftrightarrow x_1) + \\ \sum_{n=1}^{m_0} L(x_0 \to x_1^n)\overline{G}(x_0 \leftrightarrow x_1^n, x_1) \end{array} \right) \\ \times \prod_{i=1}^{k-1} \left(\begin{array}{l} f_s(x_{i-1} \to x_i \to x_{i+1})G(x_i \leftrightarrow x_{i+1}) + \\ \sum_{n=1}^{m_i} f_s(x_{i-1} \to x_i \to x^n_{i\leftrightarrow i+1})\overline{G}(x_i \leftrightarrow x^n_{i\leftrightarrow i+1}, x_{i+1}) \end{array} \right) \\ \times W(x_{k-1} \to x_k).$$

References

1. Barzel, R.: Lighting controls for computer cinematography. J. Graph. Tools **2**(1), 1–20 (1997)
2. Birn, J.: Digital Lighting and Rendering, 2nd edn. New Riders Publishing, Thousand Oaks (2005)
3. Blender Online Community: Blender—a 3d modelling and rendering package (2015). http://www.blender.org
4. Damez, C., Slusallek, P., Walter, B.J., Myszkowski, K., Wald, I., Christensen, P.H.: Global illumination for interactive applications and high-quality animations. SIGGRAPH 2003 Course Note #27, ACM, pp. 27–31. San Diego, USA (2003)

5. Hachisuka, T., Ogaki, S., Jensen, H.W.: Progressive photon mapping. ACM Trans. Graph. **27**(5), 130:1–130:8 (2008)
6. Heckbert, P.S.: Adaptive radiosity textures for bidirectional ray tracing. SIGGRAPH Comput. Graph. **24**(4), 145–154 (1990)
7. Jakob, W.: Mitsuba renderer (2010). http://www.mitsuba-renderer.org
8. Jensen, H.W.: A practical guide to global illumination using ray tracing and photon mapping. In: ACM SIGGRAPH 2004 Course Notes, SIGGRAPH '04. ACM, New York (2004)
9. Kerr, W.B., Pellacini, F., Denning, J.D.: Bendylights: artistic control of direct illumination by curving light rays. Comput. Graph. Forum **29**(4), 1451–1459 (2010)
10. Lafortune, E.P., Willems, Y.D.: Bi-directional path tracing. In: Proceedings Conference on Computational Graphics and Visualization Techniques, pp. 145–153 (1993)
11. Lagae, A., Dutré, P.: An efficient ray-quadrilateral intersection test. J. Graph. Tools **10**(4), 23–32 (2005)
12. Mattausch, O., Igarashi, T., Wimmer, M.: Freeform shadow boundary editing. Comput. Graph. Forum **32**, 175–184 (2013)
13. Nowrouzezahrai, D., Johnson, J., Selle, A., Lacewell, D., Kaschalk, M., Jarosz, W.: A programmable system for artistic volumetric lighting. ACM Trans. Graph. **30**(4), 29:1–29:8 (2011)
14. Obert, J., Pellacini, F., Pattanaik, S.: Visibility editing for all-frequency shadow design. In: Proceedings of the 21st Eurographics Conference on Rendering, EGSR'10 (2010)
15. Okabe, M., Matsushita, Y., Shen, L., Igarashi, T.: Illumination brush: interactive design of all-frequency lighting. In: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications, PG '07, pp. 171–180. IEEE Computer Society, New York (2007)
16. Pellacini, F.: Envylight: an interface for editing natural illumination. ACM Trans. Graph. **29**(4), 34:1–34:8 (2010)
17. Pellacini, F., Battaglia, F., Morley, R.K., Finkelstein, A.: Lighting with paint. ACM Trans. Graph. **26**(2) (2007)
18. Pellacini, F., Tole, P., Greenberg, D.P.: A user interface for interactive cinematic shadow design. ACM Trans. Graph. **21**(3), 563–566 (2002)
19. Raymond, B., Guennebaud, G., Barla, P., Pacanowski, R., Granier, X.: Optimizing BRDF orientations for the manipulation of anisotropic highlights. Comput. Graph. Forum **33**(2), 313–321 (2002)
20. Reiner, T., Kaplanyan, A., Reinhard, M., Dachsbacher, C.: Selective inspection and interactive visualization of light transport in virtual scenes. Comput. Graph. Forum **31**(2pt4), 711–718 (2012)
21. Ritschel, T., Okabe, M., Thormählen, T., Seidel, H.P.: Interactive reflection editing. ACM Trans. Graph. **28**(5), 129:1–129:7 (2009)
22. Ritschel, T., Thormählen, T., Dachsbacher, C., Kautz, J., Seidel, H.P.: Interactive on-surface signal deformation. ACM Trans. Graph. **29**(4), 36:1–36:8 (2010)
23. Schmidt, T-.W., Novák, J., Meng, J., Kaplanyan, A.S., Reiner, T., Nowrouzezahrai, D., Dachsbacher, C.: Path-space manipulation of physically-based light transport. ACM Trans. Graph. **32**(4), 129:1–129:11 (2013)
24. Veach, E.: Robust monte carlo methods for light transport simulation. Ph.D. thesis, Chap. 4,8, Stanford, CA, USA (1998)

Thomas Subileau is Ph.D. student in 3rd year at IRIT, University of Toulouse. His research is focused on the artistic control and editing of rendering.



Nicolas Mellado is a postdoctoral researcher at IRIT, University of Toulouse. His research interests include point cloud processing, multiscale analysis and registration. Dr. Mellado received a Ph.D. degree in Computer Science from the University of Bordeaux, France, December 2012.

David Vanderhaeghe is an Associate Professor at IRIT-Université de Toulouse where he started in 2010. He completed his Ph.D. in 2008 at Université de Grenoble-INRIA Rhône-Alpes under the supervision of Joëlle Thollot and François X. Sillion. Then he spent two years as a Postdoctoral Researcher at LaBRI-INRIA Bordeaux Sud-Ouest/Université Bordeaux 1. His research focuses on image synthesis, stylization and user control for image and animation creation. He his a member of the Eurographics short paper program committee. He is a regular reviewer for major journals and international conferences (ACM Siggraph TOG, Eurographics, IEEE-TVCG, CGF).

Mathias Paulin is Professor in Computer Science at the University Paul Sabatier of Toulouse since 2007. Researcher in the Computer Graphics group at the Institut de Recherche en Informatique de Toulouse Head of the Images and Multimedia master program at the Université Paul Sabatier.