Eurographics Symposium on Rendering 2015
J. Lehtinen and D. Nowrouzezahrai
(Guest Editors)

*Volume 34 (2015), Number 4*

# Path-space Motion Estimation and Decomposition for Robust Animation Filtering

Henning Zimmer[1] Fabrice Rousselle[1] Wenzel Jakob[2] Oliver Wang[1] David Adler[3]
Wojciech Jarosz[1] Olga Sorkine-Hornung[2] Alexander Sorkine-Hornung[1]

[1]Disney Research Zurich [2]ETH Zurich [3]Walt Disney Animation Studios

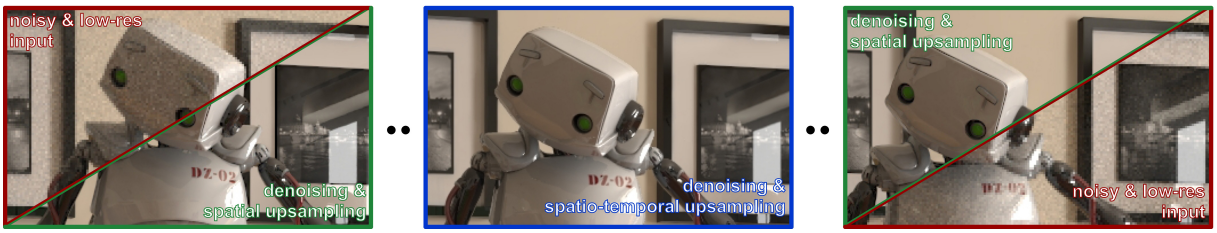



**Figure 1:** *Starting from noisy, low-resolution frames generated with a path tracer (red borders), our method improves quality and reduces computational cost by computing spatially and temporally upsampled and denoised frames (green and blue borders) while properly preserving view-dependent shading effects like the reflections in the picture frame and on the robot.*

**Abstract**

*Renderings of animation sequences with physics-based Monte Carlo light transport simulations are exceedingly costly to generate frame-by-frame, yet much of this computation is highly redundant due to the strong coherence in space, time and among samples. A promising approach pursued in prior work entails subsampling the sequence in space, time, and number of samples, followed by image-based spatio-temporal upsampling and denoising.*

*These methods can provide significant performance gains, though major issues remain: firstly, in a multiple scattering simulation, the final pixel color is the composite of many different light transport phenomena, and this conflicting information causes artifacts in image-based methods. Secondly, motion vectors are needed to establish correspondence between the pixels in different frames, but it is unclear how to obtain them for most kinds of light paths (e.g. an object seen through a curved glass panel).*

*To reduce these ambiguities, we propose a general decomposition framework, where the final pixel color is separated into components corresponding to disjoint subsets of the space of light paths. Each component is accompanied by motion vectors and other auxiliary features such as reflectance and surface normals. The motion vectors of specular paths are computed using a temporal extension of manifold exploration and the remaining components use a specialized variant of optical flow. Our experiments show that this decomposition leads to significant improvements in three image-based applications: denoising, spatial upsampling, and temporal interpolation.*

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

## 1 Introduction

The recent widespread adoption of physically based light transport techniques in the animation and visual effects industries [KKG*14] has led to tremendous computational challenges. Given that hundreds of thousands of frames are needed for a feature-length film with per-frame computation times in the order of several hours, rendering costs have become a critical bottleneck, which will continue to increase as home and cinema displays continue to move to steadily higher resolutions and frame rates.

At the same time, the underlying computation is characterized by a high amount of redundancy due to the coherence in space, time and among samples. Many acceleration techniques build on this property by using a reduced number of

samples, followed by image-based post-processing to fill in missing information. Examples include using fewer samples per pixel and denoising, leaving out some of the frames and interpolating them temporally, or rendering a subset of pixels and performing spatial upsampling. A key advantage of image-based methods is that their computational complexity scales with the number of pixels and not with the scene and lighting complexity. Consequently, such methods have become a vital part of production rendering pipelines [God14].

Unfortunately, filtering in image space is inherently ill-posed. Prior works have investigated ways of disambiguating the problem by exporting auxiliary data from the rendering process, e.g. motion vectors or depth values. This has been leveraged for real-time frame interpolation [BMS*12], spatio-temporal upsampling [HEMS10], as well as for denoising path-traced renderings [SD12]. However, when processing images produced by physically based light transport simulations, filtering remains ill-posed even with such auxiliary data: the problem is that the color of a pixel becomes a composite of various light transport effects, including the reflectance of objects, shadows, and specular reflections and refractions. These effects generally follow different trajectories over time, making it impossible to assign a single motion vector per pixel. Thus, averaging neighboring pixels for denoising or upsampling inevitably leads to blurring, ghosting and other undesirable interference.

**Contributions.** To address these issues, we propose a general decomposition framework, where the final pixel color is separated into components corresponding to disjoint subsets of path space [Vea98]. Such decompositions have been used for different applications (Section 2); the key innovation of our approach is that each component is accompanied by matching motion vectors, which makes it suitable for a range of useful filtering applications. We also show how to extract component-specific auxiliary features and propose a novel way of decomposing reflectance and incident illumination that significantly improves on prior work.

Computing motion vectors for higher order light paths requires a new set of tools: for specular paths, we propose a space-time extension of Manifold Exploration (ME) [JM12]. For the remaining components of the decomposition, we use an image-based optical flow method, bootstrapped with the motion vectors of the visible surface points.

We demonstrate the impact of our decomposition and motion estimation framework in three applications and combinations thereof: denoising, spatial upsampling, and temporal interpolation. As our framework successfully resolves the ambiguity problems discussed above, we obtain significant improvements in image quality compared to baseline methods using non-decomposed image data. Our method thus effectively reduces computational costs for producing high quality renderings of animation sequences.

## 2 Related Work

**Decompositions.** Several prior works compute sets of feature buffers which capture certain aspects of the underlying light scattering process, for instance to emphasize specular reflections in the context of non-photorealistic rendering [ST90]. The NVIDIA Iray rendering system [NVI12] uses a regular expression-based decomposition of path space to emphasize certain aspects of the light transport for artistic control. Dąbała et al. [DKR*14] decompose images into many components using Whitted-style ray trees to control stereoscopic disparity. Our construction is designed for a fundamentally different purpose: to capture and decompose animated multiple scattering effects so that they do not interfere when frames are further processed using image-based methods. Related to this idea, Lochmann et al. [LRR*14] separate diffuse reflections from specular reflections and refractions for novel view generation. However, their image space approach is restricted to a single specular bounce, whereas we propose a more fine-grained decomposition that considers arbitrary chains of specular effects.

Specialized decompositions are also used in a variety of rendering algorithms: a seminal example is the irradiance caching algorithm of Ward et al. [WRC88], which exploits the smooth nature of indirect illumination to reconstruct irradiance values from a sparse set of samples. An important observation of the irradiance caching algorithm is that the reconstruction becomes considerably easier once the effects of texture variation are separated from the incident irradiance. Irradiance factorization is commonly used in global illumination methods, e.g. for interactive rendering [WKB*02, SIMP06] or factorized axis-aligned filtering [MYRD14]. A fundamental problem of this approach is that it builds upon a linear shading model, which is invalidated by most realistic shading models [HMD*14], spatial anti-aliasing, and distribution effects such as depth-of-field or motion blur. We propose a generalized lighting–texture factorization that does not suffer from this problem; we demonstrate its effectiveness on depth-of-field and state-of-the-art microfacet reflectance models.

**Motion estimation.** Rendering engines can typically provide motion vectors for the scene geometry, which for example has been exploited for rendering novel viewpoints [MMB97]. However, these motion vectors do not capture apparent motion caused by secondary effects like shadows or reflections. In the context of filtering for anti-aliasing, Shinya [Shi95] and Igehy [Ige99] proposed techniques that can handle specular interactions as well as moving shadows. For general motion estimation, one could also resort to image-based techniques like optical flow [BBPW04] to estimate the apparent motion in the rendered images, but even state-of-the-art methods do not achieve the accuracy needed for highly accurate interpolation results. In particular, most methods only generate a single flow vector per pixel, which cannot model different motions of overlapping secondary effects. To address the problem of multiple motion vectors per pixel,

Lochmann et al. [LRR*14] compute approximate motion vectors for diffuse reflections as well as specular reflections and refractions in image space, which produces significant artifacts when used for interpolating novel viewpoints, since the continuous space-time behavior of the scene geometry is lost when operating in image space. In contrast, our motion estimation based on Manifold Exploration supports arbitrary chains of specular interactions and yields pixel-accurate motion vectors by performing computations at render time, where the 3D scene data is still available.

**Denoising.** There has been a strong renewed interest in denoising Monte Carlo renderings recently and we refer the reader to the survey by Zwicker et al. [ZJL*15] for a complete overview. Many recent methods resort to generic image space filters, which have proven to be surprisingly effective [SD12], especially when performing joint filtering guided by auxiliary features, e.g. normals and reflectance. While normals capture geometric detail and reflectance captures the appearance of materials, they miss the aforementioned secondary effects like shadows and reflections. To address these limitations, additional features have been proposed, such as a virtual flash image to capture reflections [MJL*13], or a direct visibility buffer to capture direct shadows [RMZ13]. Instead of introducing additional features, we process each component separately while using a component-specific set of auxiliary features. This not only preserves secondary effects, but also allows denoising to adapt to the noise characteristics of each component, leading to significant quality improvements.

**Spatial upsampling.** To avoid the edge blurring of straightforward spatial upsampling techniques like bilinear interpolation, joint upsampling schemes have been proposed that leverage edge information from a high-resolution guide image [HST13]. In the rendering context, such guide images are readily obtained from cheap auxiliary features like depth or normals [HEMS10]. We build on the approach of He et al. [HST13] and show that upsampling each component independently using the respective features as guides improves results in the presence of complex secondary effects.

**Temporal upsampling.** There is a large body of work on image interpolation in video or for virtual view generation [CW93, ZKU*04]. More recent work uses motion vectors to drive image warping for upsampling rendered footage for higher frame rate displays [DER*10]. However, such image-based methods are prone to visual artifacts due to unavoidable errors in image correspondence estimation and ambiguities caused by the typical restriction to a single motion vector or depth value per pixel. Frame interpolation can also be seen as a special case of the more general concept of leveraging temporal coherence, which has a long history in rendering [SYM*11]. In contrast to frame interpolation, these methods reuse shading information over time to bootstrap the rendering of frames, as opposed to synthesizing them. This produces approximate results at interactive rates, but mostly does not meet the high quality standards of offline rendering.

## 3 Decomposition

The goal of our decomposition is the separation of several different scattering effects like chains of specular reflections and refractions so that they do not interfere in image-based post-processing methods (denoising, spatial upsampling, and frame interpolation; see Section 5). To this end, we decompose the rendered image into disjoint path space components using a tree of regular expressions, where each leaf node corresponds to an image buffer (Figure 2 and 3).

We use a decomposition based on light path analysis following Veach's [Vea98] path-space formalism, which models light transport using integrals over *light paths*. Each path is a piecewise linear trajectory between the camera and a light source, where each vertex represents an intermediate scattering interaction. In order to define our decomposition, we rely on Heckbert's [Hec90] regular expression notation to classify light paths or families of similar light paths. The first vertex on the camera sensor is labeled `E`, and each subsequent scattering vertex encodes the underlying material: diffuse (`D`), specular or glossy reflection (`R`), and specular or glossy transmission (`T`). We classify glossy light interactions (e.g. scattering off rough metal or glass) as `R` or `T` if the roughness is below a threshold (Beckmann roughness $\alpha < 0.1$ in our experiments), otherwise we consider them to be diffuse. Families of similar light transport paths can be expressed using a regular expression syntax, as shown in the legend of Figure 3 and detailed in the supplementary material.

Many different decompositions are theoretically feasible and it is important to find the right trade-off: decompositions with few components may not provide much benefit, whereas very detailed path classifications provide a rich source of information, but the storage and computational overhead eventually becomes prohibitive due to the exponential growth of classes with increasing path length. Our decomposition is flexible in this regard and can be customized to focus on the salient light transport effects in specific scenes. We found the decomposition shown in Figures 2 and 3 to be adequate for most cases, but it is possible to enrich it with additional path types. In the *Vase* scene (Figure 13), we additionally track `ETTTTD`$.\star$ paths, i.e. a sequence of four specular refractions showing the distorted background behind the vase.

Each component is associated with a color buffer, and the pixel-wise sum of these buffers yields the final image. The residual component $\perp\!\!\!\perp$ captures all unmatched paths and usually contains only low-frequency, low-magnitude content. However, it is still important for the final image synthesis.

**Texture/lighting separation.** We further decompose, similarly to previous work [WRC88], the individual components into irradiance and reflectance, to separate texture from lighting (Figure 2, columns two and three). This separation is beneficial for image-based methods as texture and lighting exhibit fundamentally different characteristics in structure, noise, and motion. For instance, the motion of the reflectance buffer is determined by the motion of shaded objects, whereas

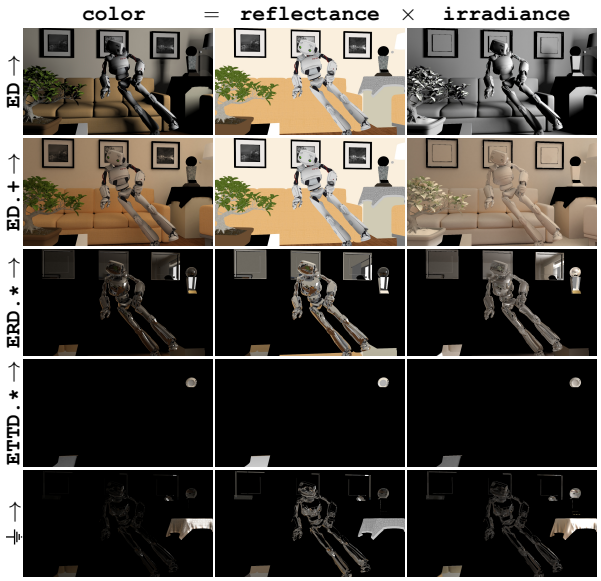



**Figure 2:** *Decomposition of the* Robot *scene from Figure 1, using the regular expression notation illustrated in Figure 3. The components (from top to bottom) are: direct diffuse (`ED`), indirect diffuse (`ED.+`), specularly reflected (`ERD.*`), specularly transmitted (`ETTD.*`), and a residual (⏚) that subsumes all previously unmatched paths. For each component we extract the color (first column), as well as the reflectance (second column), and compute the effective irradiance (third column) as the ratio between them. The final image is obtained by multiplying each component's irradiance by its reflectance, and adding up the resulting colors.*

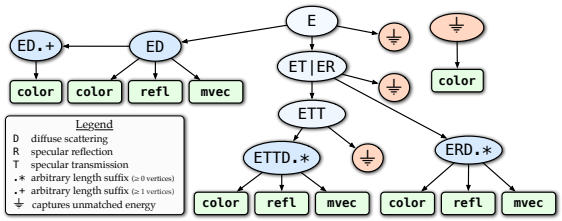



**Figure 3:** *A schematic illustration of the decomposition from Figure 2, showing its tree-like structure. As rays are traced from the camera/eye ("`E`"), their contribution and auxiliary data is stored into one of several image buffers (green).*

the apparent motion of the irradiance is also affected by the motion of occluders and light sources.

We now detail the texture/lighting separation for the direct diffuse component, but the approach can be trivially generalized to all other components. The direct diffuse component's observed color can be expressed as:

$$\texttt{ED}\rightarrow\texttt{color} = \int_{S^2} \rho(\omega_i,\omega_o) L_d(\omega_i)\,\mathrm{d}\omega_i^{\perp},$$

where we integrate the product of the BSDF $\rho(\omega_i,\omega_o)$ and the direct incident radiance $L_d(\omega_i)$ over the space of projected solid angles $\omega_i^{\perp}$ for the outgoing direction $\omega_o$. As mentioned previously, we additionally output a reflectance buffer during rendering which contains a Monte Carlo estimate of

$$\texttt{ED}\rightarrow\texttt{reflectance} = \rho(\omega_o) = \frac{1}{\pi}\int_{S^2} \rho(\omega_i,\omega_o)\,\mathrm{d}\omega_i^{\perp}.$$

In the case of a perfectly diffuse Lambertian surface, $\rho(\omega_o)$ is the standard directionless reflectance $\rho$ of the surface, but this expression nicely generalizes to an "effective reflectance" for non-Lambertian materials containing glossy transmission or reflection. Previous works then compute the irradiance as the integral of all incoming radiance $L_d(\omega_i)$ and reconstruct the component's color as the product of reflectance and irradiance. However, this breaks down whenever the product of the integrals of reflectance and irradiance over a pixel differs from the integral of the products, which occurs in the presence of non-Lambertian surfaces, distribution effects such as depth-of-field or simple spatial anti-aliasing. Unlike previous work, we do not directly evaluate the irradiance, but instead compute an "effective" irradiance as the ratio between the component's color value and the effective reflectance,

$$\texttt{ED}\rightarrow\texttt{irradiance} = \frac{\texttt{ED}\rightarrow\texttt{color}}{\texttt{ED}\rightarrow\texttt{reflectance}}.$$

As very low reflectance values lead to numerical instability we do not divide by the reflectance when it is below $10^{-3}$ and instead directly use the radiance as irradiance, which can be done as these light paths do not contribute measurably.

This effective irradiance factorization circumvents the limitations of the standard irradiance factorization (see Figure 11 for an illustration) by enforcing a linear shading model on all pixels. This ensures, by construction, that the component's color can always be reconstructed as the product of reflectance and effective irradiance.

**Features.** We also associate each component with a set of features that can tangibly improve the performance of image-based methods [HEMS10, SD12]. This data is collected at the first non-specular vertex of each path and can thus be easily obtained as a byproduct of path tracing. We extract reflectance (as mentioned before), normal, object ID, face ID, texture coordinates, and emitted radiance from visible light sources (see Figure 9 for examples). The features capture most of the characteristics of the corresponding components, including the details of the geometry seen through reflection or refraction.

We implemented our decomposition in the Mitsuba renderer [Jak10], as detailed in the supplementary material. The results shown in Section 5 heavily use Mitsuba's (smooth) `plastic` and microfacet-based `roughplastic` materials, which simulate a coated diffuse layer with a configurable Fresnel transmission and a nonlinear dependence on albedo. Similar coupled specular-matte materials are also used in

ED→reflectance (primary) motion vectors

**Figure 4:** *The diffuse reflectance component and its (forward) motion vectors—which we refer to as* primary *motion vectors—color coded as shown in the top right inset.*

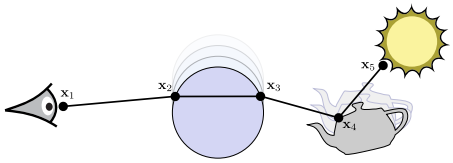



**Figure 5:** *A moving non-specular object observed through a pair of moving specular refractions. To compute the effective motion of the object in the rendered image, we fix vertices* $\mathbf{x}_1$ *and* $\mathbf{x}_4$ *of the light path and perform an implicit solve for the path configuration at the next frame.*

industrial rendering systems [HMD*14]. However, their behavior cannot be handled with prior albedo-irradiance decompositions and requires our effective irradiance factorization.

## 4 Motion Estimation

Image-based methods such as frame interpolation and temporally stable denoising require accurate motion vectors for each component of the decomposition. Disregarding the effects of shading and lighting, it is straightforward to extract motion vectors of visible surface positions on the scene geometry by mapping the underlying intersections forward in time and projecting the 3D motion into screen space (Figure 4). In the remainder we refer to these as *primary* motion vectors.

### 4.1 Specular Flow via Temporal Manifold Exploration

Specular motion vectors are significantly more challenging to extract due to the complex interaction of position, motion, orientation and curvature of the involved objects. In the following we show how to compute highly accurate specular motion vectors ("specular flow") directly within the renderer.

We propose a generalized version of the *manifold exploration* (ME) [JM12] technique to compute the apparent motion of objects observed through a sequence of specular reflection or refractions. ME is based on the observation that light, which undergoes specular scattering events, follows trajectories that lie on a lower-dimensional manifold of transport paths akin to configuration spaces of a mechanical system. By finding local parameterizations of this manifold, it is possible to explore it via a sequence of one or more local steps. In the original method, this was used to answer questions like: "if a 3D point seen through a static curved glass object moves, how does the corresponding observed point on the surface of the glass object shift?". By a temporal extension of the underlying manifolds, we can answer the same question for specular motion from frame to frame in general dynamic scenes. Instead of directly solving for the image-space motion, we solve a slightly more general problem: given a light path with vertices $\mathbf{x}_1,\dots,\mathbf{x}_N$ at time $t$, we evolve its configuration up to the next frame $t+1$. We can then project the differences in the two configurations onto the image plane and obtain the image-space motion vectors.

We now summarize the key ideas of our extension here and refer the reader to Jakob's thesis [Jak13] for a detailed explanation of the original technique for static scenes.

Vertex $\mathbf{x}_1$ is assumed to be a position on the aperture of the camera, and $\mathbf{x}_n$ ($n \leq N$, where $N$ denotes the total number of path vertices) is an interaction with a non-specular material that is observed through a chain of specular interactions. We are only interested in the behavior up to the first non-specular interaction or light source $\mathbf{x}_n$ (e.g. $\mathbf{x}_5$ in Figure 5) and ignore any subsequent vertices $\mathbf{x}_{n+1},\dots,\mathbf{x}_N$. Each specular interaction between $\mathbf{x}_1$ and $\mathbf{x}_n$ can be interpreted as a constraint that requires the incident and outgoing directions to satisfy a geometric relationship: in the case of a mirror, it requires the inclinations of the incident and outgoing directions to match. These constraints effectively collapse the set of contributing light paths to a lower-dimensional manifold, which can be explored using a Newton-like root-finder involving derivative computation and projection steps, which we describe in turn.

Our approach assumes the rendering system has the capability of querying the position of the path vertices over time while keeping them rigidly attached to the underlying camera, shape, or light source. Hence, given the initial vertices $\mathbf{x}_1(t),\dots,\mathbf{x}_n(t)$, we can find their future positions $\mathbf{x}_1(t+1),\dots,\mathbf{x}_n(t+1)$. Generally, this new path is not in a valid configuration anymore, i.e. it does not satisfy the laws of specular reflection or refraction everywhere. We therefore derive a correction term that attempts to bring the vertices back into a valid configuration by analyzing the geometric properties of a local first-order approximation of the manifold of valid light paths.

We assume that each vertex $\mathbf{x}_i(t)$ has linearly independent tangent vectors $\partial_u\mathbf{x}_i(t)$ and $\partial_v\mathbf{x}_i(t)$, and that its position can be differentiated with respect to time, yielding a 3D motion vector $\partial_t\mathbf{x}_i(t)$. We use these three quantities to define a Taylor approximation $\hat{\mathbf{x}}_i$ centered around the current vertex position $\mathbf{x}_i(t)$ which parameterizes the vertex on a small neighborhood in space (parameters $u,v$) and time (parameter $t$):

$$\hat{\mathbf{x}}_i(u,v,t) = \mathbf{x}_i + u\cdot\partial_u\mathbf{x}_i + v\cdot\partial_v\mathbf{x}_i + t\cdot\partial_t\mathbf{x}_i \qquad (1)$$

Our first change to the original ME derivation is the addition of the last temporal term, which introduces extra derivative terms that propagate through the subsequent steps. For brevity,

we now assume that all accented quantities are parameterized by $(u,v,t)$. Similarly to the above equation, we define an interpolated shading normal $\hat{\mathbf{n}}_i$ by replacing all occurrences of $\mathbf{x}_i$ by $\mathbf{n}_i$ and normalizing the result of the interpolation. Finally, we complete $\hat{\mathbf{n}}_i$ to an orthonormal three-dimensional frame $\{\hat{\mathbf{s}}_i, \hat{\mathbf{t}}_i, \hat{\mathbf{n}}_i\}$, where the $\hat{\mathbf{s}}_i$ is aligned with $\partial_u \mathbf{x}_i$ and where $\hat{\mathbf{t}}_i = \hat{\mathbf{n}}_i \times \hat{\mathbf{s}}_i$.

Suppose now that a specular reflection or refraction with a relative index of refraction $\eta$ takes place at vertex $\mathbf{x}_i$ ($\eta = 1$ in the case of reflection). If the vertex is in a valid specular configuration, its generalized half-direction vector [WMLT07]

$$\hat{\mathbf{h}}_i = \frac{\hat{\mathbf{x}}_{i-1} - \hat{\mathbf{x}}_i}{\|\hat{\mathbf{x}}_{i-1} - \hat{\mathbf{x}}_i\|} + \eta \frac{\hat{\mathbf{x}}_{i+1} - \hat{\mathbf{x}}_i}{\|\hat{\mathbf{x}}_{i+1} - \hat{\mathbf{x}}_i\|} \tag{2}$$

is collinear with the normal $\hat{\mathbf{n}}_i$. An equivalent way of stating this property is that the projection of $\hat{\mathbf{h}}_i$ onto the interpolated coordinate frame

$$\hat{\mathbf{c}}_i = \begin{pmatrix} \hat{\mathbf{h}}_i \cdot \hat{\mathbf{s}}_i \\ \hat{\mathbf{h}}_i \cdot \hat{\mathbf{t}}_i \end{pmatrix} \tag{3}$$

vanishes, i.e. $\hat{\mathbf{c}}_i = 0$. A subpath $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with endpoints $\mathbf{x}_1$ and $\mathbf{x}_n$ must then satisfy $n-2$ such constraints (one for each specular scattering event), which we collect and jointly write as $\hat{\mathbf{c}}(u_1, v_1, \ldots, u_n, v_n, t) = 0$, where $\hat{\mathbf{c}} : \mathbb{R}^{2n+1} \to \mathbb{R}^{2(n-2)}$. This equation describes a first-order approximation of an implicitly defined 5-dimensional manifold over light paths embedded in a $(2n+1)$-dimensional space. Of particular interest are the tangent vectors of this high-dimensional manifold, which express how infinitesimal movements of one vertex affect the rest of the specular chain; these can be obtained via a simple application of the implicit function theorem, which we discuss next.

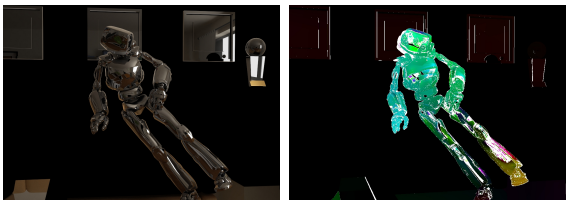
To concretely reason about tangent vectors, we must first choose coordinates in which they should be expressed. In this context, the most useful coordinates parameterize the intermediate vertex positions in terms of the endpoint positions $u_1, v_1, u_n, v_n$ (4 dimensions) and time $t$ (1 dimension). Let $J_{\hat{\mathbf{c}}}$ be the (square) Jacobian matrix of $\hat{\mathbf{c}}$ with respect to the remaining coordinates (i.e. the intermediate vertex positions):

$$J_{\hat{\mathbf{c}}} = \left( \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial u_2}, \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial v_2}, \ldots, \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial u_{n-1}}, \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial v_{n-1}} \right). \tag{4}$$

Then the desired tangent vectors are given by the columns of

$$\mathbf{T} = -J_{\hat{\mathbf{c}}}^{-1} \left( \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial u_1}, \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial v_1}, \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial u_n}, \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial v_n}, \frac{\partial \hat{\mathbf{c}}(\mathbf{0})}{\partial t} \right). \tag{5}$$

The involved derivatives are simple to evaluate using automatic differentiation. We use a dense LU factorization for the linear system solve in Eq. (5), which could be optimized to take advantage of the block tridiagonal structure of $J_{\hat{\mathbf{c}}}$ for large $n$. We did not find this necessary as $n \leq 8$ in all of our experiments ($n = 8$ was required to track quadruple refraction paths in Figure 13). Note that $n$ is only related to the depth of the decomposition, not the total path depth used in the scattering simulation.



ERD.⋆→color | specular motion vectors

**Figure 6:** *Specular reflection component and matching motion vectors found using Manifold Exploration. White pixels mark light paths which cease to exist or cannot be tracked to the next frame. When interpolating frames, we flag and recompute these pixels using a second sparse rendering phase.*

In principle, we could now use the entries of the matrix $\mathbf{T}$ to estimate motion vectors for specular paths: for instance, the fifth column specifies how the path vertices should be perturbed to maintain a valid configuration for an infinitesimal change in time. However, specular flow is highly nonlinear, and this extrapolation is not accurate enough even on a frame-to-frame basis. ME uses a sequence of alternating extrapolation and projection steps to accurately solve for path configurations; the projection step effectively re-traces the linearly extrapolated path starting from the camera $\mathbf{x}_1$, which either fails or produces a corrected light path that satisfies all specular constraints. A simple repetition of these two steps leads to a Newton-like method with quadratic convergence close to the solution. As with standard Newton methods, it is helpful to use an adaptive step size criterion to ensure that the linearity assumption is sufficiently satisfied (please refer to Jakob and Marschner [JM12] for details). On a typical frame of our *Robot* scene, the average number of spatial and temporal iterations were 3.72 and 1.37, respectively.

In our case we pursue two simultaneous goals that are different from the original method: we want to evolve a light path $\mathbf{x}_1, \ldots, \mathbf{x}_n$ from time $t$ to $t+1$ such that the endpoints $\mathbf{x}_1$ and $\mathbf{x}_n$ remain firmly attached to the underlying objects. We achieve this using two nested solves: the inner loop is a standard (i.e. non-temporal) manifold walk invoked at time $t < t' \leq t+1$ to ensure that the endpoint vertices are at their target positions $\mathbf{x}_1(t')$ and $\mathbf{x}_n(t')$. The outer loop is a temporal manifold walk, which advances the path forward in time and ensures that it remains in a valid configuration (though the endpoints may shift). Combined, they lead to a final set of positions at time $t+1$ which allows us to evaluate the change in position of the first vertex as seen from the camera, i.e. $\mathbf{x}_2(t+1) - \mathbf{x}_2(t)$, and project it into image space to obtain the final motion vector $\mathbf{v}_t(\mathbf{p})$, where $\mathbf{p} \in \Omega$ denotes a pixel position in the 2D image domain $\Omega$.

The entire process is fast compared to Monte Carlo rendering, since only a few rays need to be traced per pixel (Table 1). Our approach results in highly accurate motion vectors with reprojection errors on the order of machine precision whenever a light path could be successfully tracked from one frame

keyframe 1 warped keyframe 2 optical flow

**Figure 7:** *Optical flow-based motion estimation for the irradiance component (left). Using the primary motion vectors (Figure 4), we perform a motion compensation by warping the second keyframe (middle) where white pixels mark occlusions. The resulting flow (right) is added to the primary motion vectors to obtain the final motion estimate.*

to the other. We flag light paths that could not be tracked or that do not exist in one of the frames (Figure 6), so that image-based methods can treat them accordingly, e.g. by recomputing the associated pixels in a final sparse rendering pass following frame interpolation.

### 4.2 Image-based Irradiance and Residual Flow

Motion in the irradiance components is the result of time variation in a complex multiple scattering process. For the residual component, motion vectors are equally challenging to compute within the renderer due to the large variety of averaged path space components. For both of these components we therefore resort to image-based optical flow to estimate motion vectors. We base our approach on the well-established method of Brox et al. [BBPW04] and propose several extensions to improve accuracy in our application scenario.

Most importantly, we need to handle large displacements due to fast object or camera motion, which are known to degrade the robustness of optical flow estimation. To this end we leverage the primary flow to bootstrap the flow computation via a *motion compensation*, as illustrated in Figure 7. Here, the primary motion vectors are used to warp the second frame towards the first frame by performing a backward lookup, which compensates for camera and object motion. The remaining motion is due to secondary effects such as moving shadows, and can be robustly estimated using optical flow. Adding these motion vectors to the primary motion vectors then gives the final motion vectors between the frames.

We also detect occlusions by a forward-backward consistency check that tests whether by following the (forward) motion vector and its corresponding backward motion vector one ends more than 0.1 pixels away from the original position. Here, we cannot hope to find a matching pixel and disable the data matching term in the optical flow solver, resulting in a smooth filling-in of motion vectors due to the smoothness assumptions imposed in the optical flow formulation. See the supplementary material for more details.

For the irradiance components, we perform the optical flow computation in the logarithmic domain, where shadows at different brightness levels can be better distinguished.

**Figure 8:** *We detect silhouettes in the scene and mark them to be ignored by image-based methods since they contain competing motion vectors.*

### 4.3 Silhouette and Occlusion Handling

Silhouette pixels capture light from both foreground and background which makes it impossible to define a single unique motion vector there. We hence detect silhouettes (Figure 8) and ignore the corresponding motion vectors for image-based methods, i.e. we treat them like untracked motion vectors in the specular flow (Figure 6). We find silhouettes as sign differences in the dot product of adjacent faces with the direction to the camera and rasterize these as 3D lines. This is very efficient, taking 6s at 1280x720 pixels.

A similar reasoning applies to occlusions, where image-space motion vectors would describe the motion of the occluded background points, which leads to artifacts when used for image-based methods. We hence detect occlusions as mentioned before and also ignore motion vectors there.

## 5 Applications

We now show how our decomposition and motion vectors can benefit different image-based post-processing methods: denoising of low sample-count renderings (Section 5.1), spatial upsampling to a higher resolution (Section 5.2), and temporal upsampling, i.e. interpolation of in-between frames (Section 5.3) to increase the frame rate.

### 5.1 Denoising

For denoising we build upon the joint NL-means filtering approach of Rousselle et al. [RMZ13], which computes the denoised value $\hat{u}(\mathbf{p})$ of a pixel $\mathbf{p}$ as a weighted average over a square neighborhood $\mathcal{N}(\mathbf{p})$ : $\hat{u}(\mathbf{p}) = \sum_{\mathbf{q}\in\mathcal{N}(\mathbf{p})} w(\mathbf{p},\mathbf{q})\,u(\mathbf{q})$, where $w(\mathbf{p},\mathbf{q}) = \min(w_c(\mathbf{p},\mathbf{q}), w_f(\mathbf{p},\mathbf{q}))$ combines a color weight $w_c$ computed on the color buffer and a feature weight $w_f$. If multiple features are available (as in our case), then $w_f$ is the minimum of the feature weights. A neighboring pixel $\mathbf{q}$ is therefore given a high weight only if it is similar to $\mathbf{p}$ according to the color and each feature.

We can directly leverage our decomposition for a joint NL-Means denoising since we render, for each component, the final color, as well as the reflectance, normal, and object ID features. We separately denoise the effective irradiance, computed as described in Section 3, of each component. The

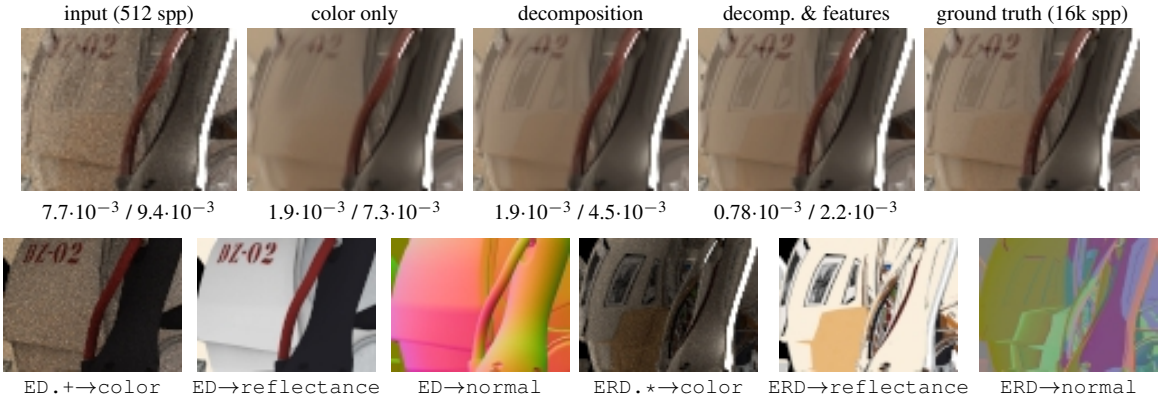



**Figure 9:** *Denoising is significantly more robust when leveraging our decomposition. Top row: NL-Means filtering the final color ("color only") yields a smooth but blurry output; NL-Means filtering each component separately ("decomposition") yields sharper reflections, but low-constrast texture details, such as the text printed on the torso, are still problematic; joint NL-Means filtering of each component guided by our auxiliary features ("decomp. & features") robustly recovers fine details in the scene and yields a result close to the ground truth. We give the relative MSE of each image for the full frame (first value) and the crop shown (second value). Second row: noisy data and corresponding reflectance and normal buffers for the indirect diffuse component (left three images) and the specular reflection component (right tree images), showing how the decomposition succeeds in capturing the distinct noise characteristic and spatial structure of the previously composited shading effects.*

color weight $w_c$ is computed on the irradiance buffer, and the feature weight $w_f$ is computed on the normal and object ID buffers. Once the irradiance is denoised, we multiply back the corresponding reflectance buffer to obtain the denoised component color. All denoised components are finally summed up to yield the denoised rendering.

In the work of Rousselle et al., a filter bank of three joint NL-Means filters was computed and combined on a per-pixel basis. One filter was more sensitive to color differences, one more sensitive to feature differences, and a third one more balanced. In our implementation, we opted to use a single balanced filter. Combined with our decomposition, we found this simplified denoising scheme offered a better trade-off between quality and complexity (see Figures 9 and 10 for results using both our implementation and the original scheme proposed by Rousselle et al.).

**Temporal denoising.** Residual low-frequency noise can lead to highly noticeable large-scale flickering in denoised animation sequences. This problem can be alleviated by a spatio-temporal filtering [HEMS10]. Extending a joint NL-Means filter to the spatio-temporal domain is easily achieved by augmenting the filtering window to include data from temporally adjacent frames [BCM08]. However, one needs to account for camera and scene motion to be able to leverage the coherence from frame to frame in an optimal way. To this end, we warp every component of adjacent frames, as well as the corresponding feature buffers, using the computed per-component motion vectors, which aligns them to the current frame. This works in the same manner as for the motion compensation described in Section 4.2. When denoising the irradiance, we use the computed geometry motion vectors of the corresponding component (`ED.*`, `ERD.*`, or `ETTD.*`), where the motion of moving shadows is not captured over time. However, the robust NL-Means weights ensure that shadows are not excessively blurred despite the misalignment. From the denoised irradiance components we can then compute motion vectors, as described in Section 4.2, to be used in other applications, such as frame interpolation. The results of spatio-temporal filtering across 3 frames are included in our supplementary video.

**Results.** In Figure 9 we compare different denoising results leveraging progressively more information from our decomposition. Using our decomposition already recovers much more details compared to denoising the final colors only. When adding the features for a joint NL-Means filtering, we achieve denoising results that are visually very close to a high sample-count ground truth rendering with a low relative mean-square error (MSE). See our supplementary material for the parameters used for the joint NL-Means filter, results at lower sampling rates, and an equal error rendering without denoising.

We also applied the original denoising algorithm proposed by Rousselle et al. [RMZ13] on our *Robot* scene, to illustrate how an advanced denoising scheme can benefit from our decomposition. The results are given in Figure 10 and show a significant improvement. Given that the choice of denoising method is orthogonal to our decomposition, we would expect other methods to benefit similarly.

**Spatial anti-aliasing and distribution effects.** Our effective irradiance factorization can be used in cases where the standard irradiance factorization is invalid, as explained

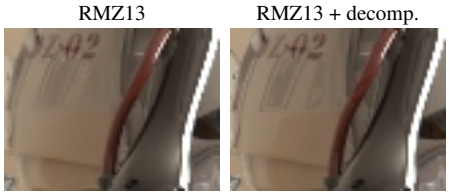



**Figure 10:** *Denoising the robot scene using the original method proposed by Rousselle et al. [RMZ13] (left image), and the same method applied on top of our decomposition (right image). Even with this more advanced method, our decomposition brings a significant improvement, both qualitatively and quantitatively, as measured by the relative MSE (which we computed on the full frame).*

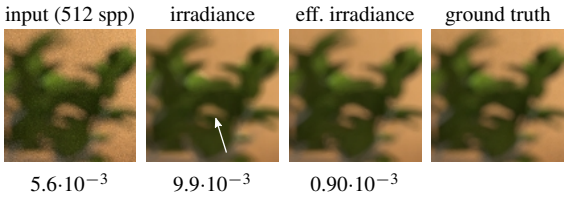



**Figure 11:** *Denoising of the* Robot *scene using irradiance factorization with added depth-of-field. Although this crop shows only diffuse materials, the linear shading assumption at the core of the irradiance factorization is invalidated by the coupling between irradiance and reflectance induced by the depth-of-field effect (and to a lesser extent the spatial anti-aliasing). Denoising using standard irradiance factorization (second column) actually degrades the MSE because of the excessive bias (most noticeable in the shrunken region pointed by the arrow). Our effective irradiance factorization (third column) does not suffer from such issues.*

in Section 3. This notably occurs when using spatial anti-aliasing and in the presence of distribution effects such as depth-of-field or motion blur. All results presented use spatial anti-aliasing, and, in Figure 11, we illustrate the output of our denoising technique when adding depth-of-field to the *Robot* scene and compare it to the result obtained using standard irradiance factorization. Note that in this case, the features also exhibit a noticeable amount of noise which requires to prefilter them as proposed in [RMZ13].

### 5.2 Upsampling

Dramatic increases in the pixel count of cinema and home TV have made it prohibitively expensive to render animations for these new formats. This can be alleviated by rendering at a lower resolution, followed by an upsampling step that produces the desired output resolution. Similarly to the improvements we achieve for denoising, upsampling each component separately yields tangibly better results, as multiple shading

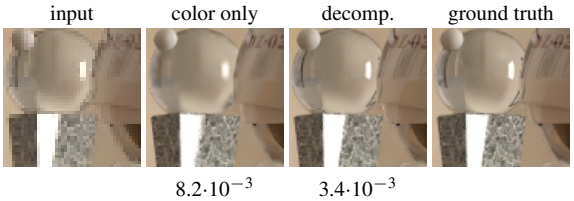



**Figure 12:** *Upsampling a 640×360 rendering at 512 spp ("input") to 1280×720 pixels. Upsampling of the final pixel color guided by the features of the diffuse component ("color only") blurs out secondary effects, such as the refractions in the sphere and the reflections on the robot. We can robustly reconstruct the reflections and refractions by applying the same upsampling scheme on top of our decomposition ("decomp."), yielding a much lower relative MSE (which we computed on the full frame). The ground truth shown is the denoised full resolution rendering at 512 spp.*

contributions that would normally interfere in the upsampling process can be disambiguated. Additionally we can leverage auxiliary features that can be cheaply computed at the high target resolution to guide the upsampling [HEMS10].

Our upsampling application is based on the guided image filter [HST13]. We use a joint upsampling scheme, where the color is upsampled using the features (reflectance, normal and emitted radiance) as a guide, and each component is upsampled individually. We directly render the image at the target high resolution but with correspondingly fewer samples. For all examples, we rendered at 1/4 of the samples. We then subsample using a 2×2 box filter, which results in a low-resolution image with reduced noise. The subsampled images are denoised, and then upsampled back to the full resolution. This reduces the sampling rate by a factor of 4, while keeping the same signal-to-noise ratio in the low-resolution image.

Due to its unconstrained optimization, the guided image filter upsampling can produce negative pixel values, particularly along strong edges. Pixels with negative values are flagged to be re-rendered with a higher sampling count and denoised at the full resolution. In practice, the re-rendering rate is very low. For the *Robot* scene it varies between 0.14% and 0.28% per frame, with an average of 0.20%.

**Results.** In Figure 12, we show that directly upsampling the final colors loses a lot of detail, despite being a common approach [SGNS07]. While it correctly reconstructs the diffuse shading, it fails for secondary effects, such as reflections and refractions. In contrast, the same upsampling scheme applied on each component of our decomposition and guided by the features yields a significantly improved result, with correct reflections and refractions, and a drastically reduced relative MSE. Please see the supplementary video for upsampling results on the full sequence.

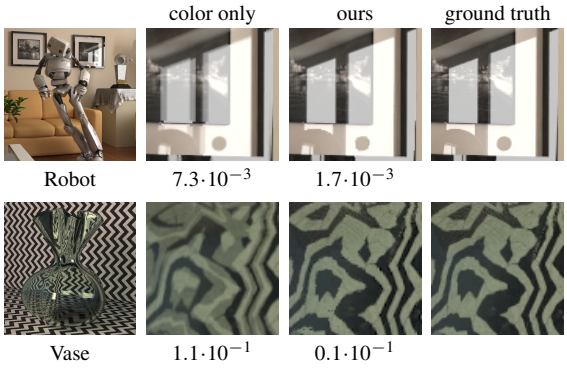



**Figure 13:** *Interpolation with challenging specular effects. The crop of the* Robot *scene (top) shows the handling of reflections (*`ERD.*`*) whereas the* Vase *scene (bottom) showcases a highly complicated fourfold specular transmission (*`ETTTTD.*`*). The* color only *baseline method (second column) interpolates the final colors using the primary motion vectors, yielding strong artifacts and a high relative MSE (which we compute for the full frame). Our approach of interpolating each component separately using the corresponding motion vectors significantly improves results. We interpolated 3 in-between frames for both scenes.*

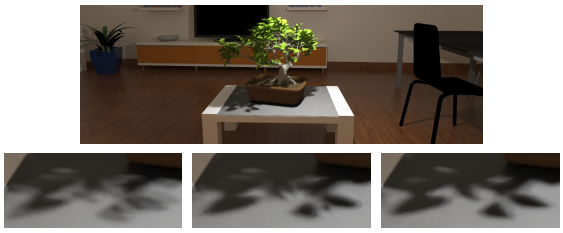



**Figure 14:** *Interpolation with challenging moving shadows. The baseline method (*color only*) leaves halos whereas our approach of interpolating the irradiance and residual components separately using the estimated motion vectors (Section 4.2) significantly reduces these artifacts and lowers the relative MSE (computed on the full frame).*

### 5.3 Frame Interpolation

Frame interpolation proceeds by projecting pixels rendered at sparse keyframes to in-between frames using corresponding motion vectors. To improve results one can use motion vectors computed forward and backward in time, and average the contributions from the two neighboring keyframes, weighted by their distance to the interpolated frame [YTS*11].

In our decomposition framework, we interpolate each component separately using the corresponding motion vectors. This remedies ghosting artifacts that appear in the the presence of complex secondary effects such as reflections and refractions where a single pixel receives contributions from various sources that might all undergo different motions.

As we compute specular motion vectors at the final frame rate, they are defined between subsequent frames. For interpolation, we however need motion vectors defined between keyframes and the current in-between frame, which we achieve by concatenating the motion vectors. The same applies to the primary motion vectors used for the diffuse reflectance component. For the irradiance and residual components, we compute motion vectors between the keyframes using optical flow (Section 4.2) and thus just scale these motion vectors w.r.t. the position of the in-between frame.

**Re-rendering.** There are several reasons why it can be necessary to re-render some pixels in the interpolated frames in a second sparse render pass: Some specular paths cease to exist or cannot be tracked over time, which leads to unknown pixels in the specular motion vectors (Section 4.1). We also ignore motion vectors of silhouette pixels as they may capture objects with conflicting motion. Such undefined motion vectors result in holes in the interpolated frame, which need to be filled by re-rendering. Holes are also caused by pixels that are occluded in both keyframes, but become visible in the in-between frames. Finally, if the illumination changes noticeably, a seam can occur between disocclusions where only a single keyframe contributes and the neighboring regions where contributions from two keyframes are averaged, making it necessary to also re-render the disoccluded region in such cases. For the *Robot* sequence, we re-render between 8% and 33% of pixels, with an average of 20%.

**Results.** In Figure 13 we show that using our decompositions and the corresponding motion vectors resolves the ghosting at specular effects that occur when interpolating the final colors only using the primary motion vectors. Figure 14 shows results for a scene with complicated moving shadows. These are captured in the diffuse irradiance and the residual components and we estimate their motion vectors using optical flow (Section 4.2). Consequently we cannot match the accuracy of the specular motion vectors obtained using Manifold Exploration, but still obtain visually appealing results.

The supplementary video shows interpolation results on full sequences where the ghosting artifacts result in strongly visible, temporal flickering.

**Glossy objects.** In the video we show additional scenes, including one featuring a highly glossy object (*Glossy Sphere*). Here, we compute specular motion vectors the same way as for ideally smooth ones, i.e. we pretend that they are smooth when running the Manifold Exploration. This is based on an empirical observation that the effective motion in the smooth and glossy case is almost identical, even though the appearance in the rendering may significantly deviate. The results in the video validate that this approximation is sufficiently accurate to obtain favorable interpolation results.

**Motion blur.** We can also use the computed motion vectors to add motion blur as a post-process by integrating (splatting)

no motion blur | added motion blur (half exposure) | added motion blur (full exposure)

**Figure 15:** *We can use our per-component motion vectors to simulate different amounts of motion blur as a post-process.*

the pixel colors of each component along the motion vectors [CW93, BE01]. As shown in Figure 15, this not only produces realistic motion blur, but also allows us to easily control the amount of blur, i.e. the virtual exposure time, after rendering. We fill pixels with undefined specular motion vectors (Figure 6) using standard image inpainting [BSCB00], which we found sufficient to obtain visually plausible results.

### 5.4 Combination

To combine the previously described image-based methods into a single pipeline we first produce high resolution keyframes by combining denoising and upsampling (Section 5.1 and 5.2). Then we interpolate in-between frames, followed by re-rendering (Section 5.3). Corresponding results are shown in Figure 1 and in the supplementary video.

## 6 Performance

Table 1 shows the speedups achieved by several combinations of our post-processing methods on the *Robot* scene, running a CPU implementation on a cluster. Storing the decomposition incurs a memory overhead proportional to its granularity. The data is stored in compressed multi-channel EXRs, leveraging that some components have many zero pixels. The frame shown in Figure 2 is 29MB in total, where the final pixel color accounts for 3.7MB.

## 7 Limitations

Our decomposition cannot fully disambiguate motion in the case of silhouette pixels overlapping regions with conflicting motions, which we handle by detecting and re-rendering these pixels. However, this heuristic is very conservative and could possibly be improved, leading to further performance gains. Similarly, our heuristic for handling occlusions is also very conservative, and it may be possible to use more advanced blending techniques, e.g., Poisson blending [PGB03] to handle occlusion seams with less re-rendering.

We also envision an automatic method to determine the components that are needed to best represent a given scene, e.g. by analyzing the materials. This could be supported by a code generation tool that automatically instruments the path tracing integrator for a given decomposition.

**Table 1:** *Average computation times per input frame (in core minutes) on the* Robot *scene (1280×720 pixels). The noisy baseline render without extracting our decomposition (***Base***) used 512 spp, whereas an (almost) noise-free ground truth render (***GT***) required 16k spp. The last three columns contrast the gain in resolution (due to upsampling or interpolation) vs. the computational overhead compared to the baseline, and give the relative cost as the ratio between the two. Clearly, as we combine more post-processing steps (***D****enoising,* ***U****psampling,* ***I****nterpolation) we reduce the relative cost. Note that the* Upsample *and* Interpolate *columns also include the re-rendering and denoising of missing pixels, which for the interpolation make up for 90% of the time.*

| **Method** | Render | Denoise | Upsample | Interpolate | **Resolution** | **Overhead** | **Rel. Cost** |
|---|---|---|---|---|---|---|---|
| **Base** | 178.0 | – | – | – | – | – | – |
| **GT** | 5696.0 | – | – | – | **1**× | **32.0**× | **32** |
| **D** | 208.2 | 5.5 | – | – | **1**× | **1.2**× | **1.2** |
| **DI** | 210.8 | 5.4 | – | 172.1 | **4**× | **2.2**× | **0.55** |
| **DU** | 61.7 | 1.5 | 12.2 | – | **4**× | **1.7**× | **0.43** |
| **DUI** | 64.3 | 1.4 | 11.1 | 131.4 | **16**× | **4.7**× | **0.29** |

Finally, we did not investigate volumetric effects in this paper. Recent efforts in the movie industry [Kai12] involve the use of "deep" buffers to capture and decompose the effects of volumetric light transport for the purpose of compositing. Extending our system with additional "deep" components would be an interesting avenue of future work.

## 8 Conclusions

We have presented a general and flexible decomposition framework for path tracing-based rendering, which resolves ambiguities that have so far hampered the adoption of image-based post-processing methods in scenarios with high quality requirements. We demonstrated the benefits of our approach for denoising, spatial upsampling and frame interpolation.

Our decomposition leverages two key contributions. First, we provide motion vectors for all components of the decomposition, including specular paths for which we derived a temporal extension of manifold exploration, and the irradiance and residual components for which we used image-based optical flow bootstrapped with the motion of visible objects. Second, we proposed a simple modification of irradiance factorization that can, once combined with our decomposition, handle general BRDFs, including specular surfaces, and challenging distribution effects such as depth-of-field.

**Acknowledgments.** We thank Maurizio Nitti for creating the *Robot* scene and Olesya Jakob for designing the vase shown in Figure 13. We also thank our colleagues at the Walt Disney Animation Studios and Industrial Light & Magic for fruitful discussions. Wenzel Jakob was supported by an ETH/Marie Curie fellowship.

## References


[BBPW04] BROX T., BRUHN A., PAPENBERG N., WEICKERT J.: High accuracy optical flow estimation based on a theory for warping. In *ECCV* (2004), pp. 25–36. 2, 7

[BCM08] BUADES A., COLL B., MOREL J.: Nonlocal image and movie denoising. *International Journal of Computer Vision 76*, 2 (2008), 123–139. 8

[BE01] BROSTOW G. J., ESSA I.: Image-based motion blur for stop motion animation. In *SIGGRAPH* (2001), pp. 561–566. 11

[BMS*12] BOWLES H., MITCHELL K., SUMNER R. W., MOORE J., GROSS M. H.: Iterative image warping. *Comp. Graph. Forum 31*, 2 (2012), 237–246. 2

[BSCB00] BERTALMIO M., SAPIRO G., CASELLES V., BALLESTER C.: Image inpainting. In *SIGGRAPH* (2000), pp. 417–424. 11

[CW93] CHEN S. E., WILLIAMS L.: View interpolation for image synthesis. In *SIGGRAPH* (1993), pp. 279–288. 3, 11

[DER*10] DIDYK P., EISEMANN E., RITSCHEL T., MYSZKOWSKI K., SEIDEL H.-P.: Perceptually-motivated real-time temporal upsampling of 3D content for high-refresh-rate displays. *Comp. Graph. Forum 29*, 2 (2010), 713–722. 3

[DKR*14] DĄBAŁA Ł., KELLNHOFER P., RITSCHEL T., DIDYK P., TEMPLIN K., MYSZKOWSKI K., ROKITA P., SEIDEL H.-P.: Manipulating refractive and reflective binocular disparity. *Comp. Graph. Forum (Proc. Eurographics 2014) 33*, 2 (2014), 53–62. 2

[God14] GODDARD L.: Silencing the noise on Elysium. In *ACM SIGGRAPH 2014 Talks* (2014), ACM. 2

[Hec90] HECKBERT P. S.: Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH* (1990), pp. 145–154. 3

[HEMS10] HERZOG R., EISEMANN E., MYSZKOWSKI K., SEIDEL H.-P.: Spatio-temporal upsampling on the GPU. In *SI3D* (2010), pp. 91–98. 2, 3, 4, 8, 9

[HMD*14] HILL S., MCAULEY S., DUPUY J., GOTANDA Y., HEITZ E., HOFFMAN N., LAGARDE S., LANGLANDS A., MEGIBBEN I., RAYANI F., DE ROUSIERS C.: Physically based shading in theory and practice. In *ACM SIGGRAPH Courses* (2014). 2, 5

[HST13] HE K., SUN J., TANG X.: Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell. 35*, 6 (2013), 1397–1409. 3, 9

[Ige99] IGEHY H.: Tracing ray differentials. In *SIGGRAPH* (1999), pp. 179–186. 2

[Jak10] JAKOB W.: Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 4

[Jak13] JAKOB W.: *Light Transport on Path-Space Manifolds*. PhD thesis, Cornell University, Aug. 2013. 5

[JM12] JAKOB W., MARSCHNER S.: Manifold exploration: A markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph. 31*, 4 (2012), 58:1–58:13. 2, 5, 6

[Kai12] KAINZ F.: Interpreting OpenEXR deep pixels, 2012. http://openexr.com/InterpretingDeepPixels.pdf. 11

[KKG*14] KRIVÁNEK J., KELLER A., GEORGIEV I., KAPLANYAN A., FAJARDO M., MEYER M., NAHMIAS J.-D., KARLÍK O., CANADA J.: Recent advances in light transport simulation: Theory and practice. In *ACM SIGGRAPH Courses* (2014). 1

[LRR*14] LOCHMANN G., REINERT B., RITSCHEL T., MÜLLER S., SEIDEL H.: Real-time reflective and refractive novel-view synthesis. In *VMV 2014* (2014), pp. 9–16. 2, 3

[MJL*13] MOON B., JUN J. Y., LEE J., KIM K., HACHISUKA T., YOON S.: Robust image denoising using a virtual flash image for monte carlo ray tracing. *Comp. Graph. Forum 32*, 1 (2013), 139–151. 3

[MMB97] MARK W. R., MCMILLAN L., BISHOP G.: Post-rendering 3D warping. In *SI3D* (1997), pp. 7–16, 180. 2

[MYRD14] MEHTA S. U., YAO J., RAMAMOORTHI R., DURAND F.: Factored axis-aligned filtering for rendering multiple distribution effects. *ACM Trans. Graph. 33*, 4 (July 2014). 2

[NVI12] NVIDIA: Iray whitepaper, 2012. http://www.nvidia-arc.com/fileadmin/user_upload/iray_2013/documents/iray_whitepaper.121206.pdf. 2

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Trans. Graph. 22*, 3 (July 2003), 313–318. 11

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Comp. Graph. Forum 32*, 7 (2013), 121–130. 3, 7, 8, 9

[SD12] SEN P., DARABI S.: On filtering the noise from the random parameters in monte carlo rendering. *ACM Trans. Graph. 31*, 3 (2012), 18. 2, 3, 4

[SGNS07] SLOAN P. J., GOVINDARAJU N. K., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *Pacific Graphics* (2007), pp. 97–105. 9

[Shi95] SHINYA M.: Improvements on the pixel-tracing filter: Reflection/refraction, shadows & jittering. In *Graphics Interface* (1995), pp. 92–92. 2

[SIMP06] SEGOVIA B., IEHL J. C., MITANCHEY R., PÉROCHE B.: Non-interleaved deferred shading of interleaved sample patterns. In *Proc. Graphics Hardware* (2006), ACM. 2

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-D shapes. *SIGGRAPH Comp. Graph. 24*, 4 (1990). 2

[SYM*11] SCHERZER D., YANG L., MATTAUSCH O., NEHAB D., SANDER P. V., WIMMER M., EISEMANN E.: A survey on temporal coherence methods in real-time rendering. In *EUROGRAPHICS 2011 State of the Art Reports* (2011). 3

[Vea98] VEACH E.: *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford, CA, USA, 1998. AAI9837162. 2, 3

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive global illumination using fast ray tracing. In *Proceedings of the 13th Eurographics Workshop on Rendering (EGWR)* (2002), pp. 15–24. 2

[WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. In *Rendering Techniques* (2007), pp. 195–206. 6

[WRC88] WARD G. J., RUBINSTEIN F. M., CLEAR R. D.: A ray tracing solution for diffuse interreflection. *SIGGRAPH Comp. Graph. 22*, 4 (1988), 85–92. 2, 3

[YTS*11] YANG L., TSE Y.-C., SANDER P. V., LAWRENCE J., NEHAB D., HOPPE H., WILKINS C. L.: Image-based bidirectional scene reprojection. *ACM Trans. Graph. 30*, 6 (2011). 10

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.: Recent advances in adaptive sampling and reconstruction for monte carlo rendering. *EUROGRAPHICS 2015 State of the Art Reports* (2015). 3

[ZKU*04] ZITNICK C. L., KANG S. B., UYTTENDAELE M., WINDER S. A. J., SZELISKI R.: High-quality video view interpolation using a layered representation. *ACM Trans. Graph. 23*, 3 (2004), 600–608. 3