DOI: 10.1111/cgf.142616
EUROGRAPHICS 2021 / N. Mitra and I. Viola
(Guest Editors)

Temporally Reliable Motion Vectors for Real-time Ray Tracing

Zheng Zeng [1], Shiqiu Liu[2], Jinglei Yang[3], Lu Wang[†1] and Ling-Qi Yan[†3],

[1]School of Software, Shandong University, China
[2]NVIDIA, USA
[3]University of California, Santa Barbara, USA

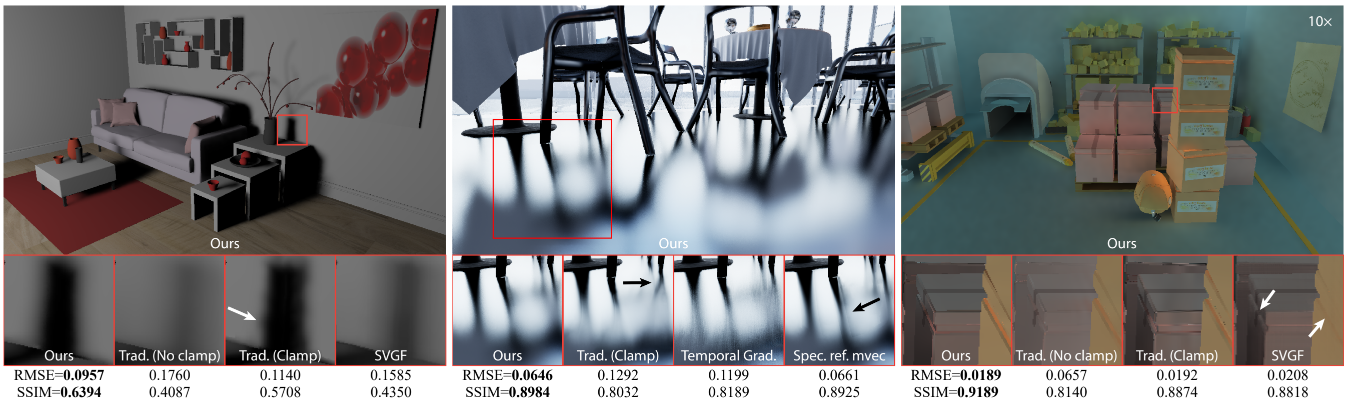
10×
Ours
Ours
Ours
Ours
Trad. (No clamp)
Trad. (Clamp)
SVGF
Ours
Trad. (Clamp)
Temporal Grad.
Spec. ref. mvec
Ours
Trad. (No clamp)
Trad. (Clamp)
SVGF
RMSE=**0.0957** 0.1760 0.1140 0.1585
SSIM=**0.6394** 0.4087 0.5708 0.4350
RMSE=**0.0646** 0.1292 0.1199 0.0661
SSIM=**0.8984** 0.8032 0.8189 0.8925
RMSE=**0.0189** 0.0657 0.0192 0.0208
SSIM=**0.9189** 0.8140 0.8874 0.8818

Figure 1: *We propose temporally reliable motion vectors for shadows (left,* pink room *scene), glossy reflections (middle,* contemporary restaurant *scene) and occlusions (right,* PICA *scene) that better explore and utilize temporal correspondences between adjacent frames. Our method significantly reduces noise, overblur and ghosting artifacts compared to the state of the art temporal reuse methods with traditional motion vectors. The quantitative evaluation metrics (RMSE and SSIM) are shown below the images.*

Abstract

Real-time ray tracing (RTRT) is being pervasively applied. The key to RTRT is a reliable denoising scheme that reconstructs clean images from significantly undersampled noisy inputs, usually at 1 *sample per pixel as limited by current hardware's computing power. The state of the art reconstruction methods all rely on temporal filtering to find correspondences of current pixels in the previous frame, described using per-pixel screen-space motion vectors. While these approaches are demonstrated powerful, they suffer from a common issue that the temporal information cannot be used when the motion vectors are not valid, i.e. when temporal correspondences are not obviously available or do not exist in theory.*

We introduce temporally reliable motion vectors that aim at deeper exploration of temporal coherence, especially for the generally-believed difficult applications on shadows, glossy reflections and occlusions, with the key idea to detect and track the cause of each effect. We show that our temporally reliable motion vectors produce significantly better temporal results on a variety of dynamic scenes when compared to the state of the art methods, but with negligible performance overhead.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Rendering; Ray tracing;***

1. Introduction

It is generally believed that ray tracing has become the gold standard in rendering to generate realistic images in a physically correct way. But ray tracing is historically slow, usually taking hours to days to produce a single noise-free image. For this reason, real-time ray tracing (RTRT) that performs at > 30 frames per second was considered impractical for a long time, until the recent breakthrough of graphics hardware brought it into life. Nowadays, real-time ray tracing has not only attracted enormous interest in the

[†] Corresponding authors: luwang_hcivr@sdu.edu.cn, lingqi@cs.ucsb.edu

video games industry, but also been pervasively explored in movies, animations, computer aided designs and visualizations for interactive previews, dynamic lighting and material editing.

Despite its rapid progress, RTRT still only allows for low sampling rates. For example, in order to guarantee real-time performance, only approximately 4 rays can be traced per pixel on the state of the art NVIDIA RTX hardware. This makes exactly 1 path sample per pixel – 1 ray each for primary hit, primary shadow, secondary hit, and secondary shadow – for all the global and local effects together. As a result, the rendered images will be noisy.

So, at the heart of the RTRT is the denoising or reconstruction technique. However, with such a low sampling rate and high performance requirement, accurate image filtering methods [MIYM15; MMMG16] are likely to fail. Processing each frame itself may take a prohibitively long time already, and it is difficult to guarantee temporal coherence to avoid flickering.

To deal with temporal stability and improve the essential sampling rate, various research directions have been explored, among which the temporal filtering stands out [CKS*17; SKW*17; MMBJ17; SPD18]. This line of work is inspired by the idea of Temporal Anti-Aliasing (TAA) [KTD*14]. They find where the shading point within each pixel was in its previous frame, and blend the previous pixel value with the current. In this way, the shading results are accumulated and smoothed over time.

Finding the correspondence of a current pixel in its previous frame is essentially calculating the optical flow [HS81], but in a much faster and more accurate *back-projection* way since every scene parameter (model, view, projection matrices over time, etc.) in rendering is known. The resulting correspondence is represented as *motion vectors*, which is a 2D vector in the image space that points from the current pixel location to its previous location in the last frame.

However, the motion vectors may not always exist. For example, a static location in the background may be blocked by a moving object in the previous frame. In this case, the motion vector does not exist at the current location. Also, the motion vectors may be wrong for effects like shadows and reflections. For example, a static shadow receiver will always have a zero-length motion vector, but the shadows casted onto it may move arbitrarily along with the light source. In any of these cases, when correct motion vectors are not available but temporal filtering is applied anyway, ghosting artifacts (unreasonable leak or lag of shading over time) will emerge.

While the temporal failures can be detected with smart heuristics [SPD18], the temporal information in these cases will be *simply rejected* nonetheless. But we believe that the information can be *better utilized*. We find and calculate different types of motion vectors for different effects, to make the seemingly unusable temporal information available again. Specifically, we introduce

- a shadow motion vector for moving shadows,
- a stochastic glossy reflection motion vector for glossy reflections, and
- a dual motion vector for occlusions.

The rest of this paper is organized as follows. In Sec. 2, we give a broader overview of related work. In Sec. 3, we briefly introduce the calculation of traditional motion vectors and their use in temporal filtering, to motivate our method. In Sec. 4, we present our temporally reliable motion vectors for each of the above mentioned effects with implementation details in Sec. 5. And in Secs. 6 and 7, we compare our method with the state of the art methods, discuss the choices / alternatives and analyze the results. In Sec. 8, we summarize our method and propose future insights.

2. Related Work

Traditional reconstruction methods. Accurate reconstruction approaches from Monte Carlo ray traced renderings were pervasively studied, but mostly focused on offline applications [EHDR11; MIYM15; MMMG16; BRM*16] that may take seconds to minutes to denoise a single image. With the development of modern hardware that allows for interactive ray tracing with very few rays per pixel, faster reconstruction methods, such as axis-aligned filtering [MWR12], fast sheared filtering [YMRD15; VMCS15] and texture space filtering [MHC*16], brought the reconstruction time to the level of tens of milliseconds per frame, thus allowing for interactive performance. While these methods explore statistics or frequency analysis of light transport to maximize the use of ray samples, the theoretical need of the number of samples are still high, and the performance is still far from real-time requirements (typically no more than 5 ms are allowed for reconstruction).

Temporal accumulation and anti-aliasing. For plausible real-time reconstruction, researchers gradually refer to temporal accumulation to increase the essential sampling rate. While a vast amount of work applies temporal accumulation [WDP99; NSL*07; KTD*14; IMK*16; PSK*16], one that has been pervasively adopted by the industry is the temporal anti-aliasing (TAA) method [KTD*14]. The key idea is to find a correspondence in the previous frame for each pixel in the current frame, then accumulate previous pixel values to the current. When the previous color differs too much from the current, a clamping operation is performed to clip the previous color closer. This reduces the ghosting artifacts but re-introduces noise and produces other artifacts such as visible cuts. In contrast, our method aims at better search of previous pixel positions where the values are close to the current, therefore reducing the noise and artifacts.

Real-time denoising methods. With the temporal accumulation, real-time denoising methods have brought RTRT into reality. Mara et al. [MMBJ17] separately filter the diffuse and glossy light transport. The recurrent autoencoder (RAE) method [CKS*17] uses recurrent connections of a neural network to simultaneously perform spatial and temporal filtering. The spatiotemporal variance-guided filtering (SVGF) method [SKW*17] analyzes both spatial and temporal variances to guide the shape / size of spatial filters, which is later extended to use an estimated temporal gradient to adaptively determine how much to rely on temporal information (A-SVGF) [SPD18]. But still unlike our method, this adaptive method does not attempt to find more reliable temporal information. Koskela et al. [KIM*19] use block-wise filtering for fast spatial reconstruction, but is completely orthogonal to temporal filtering, thus is still prone to ghosting artifacts.

Accurate motion vectors have been studied previously to find better temporal information. Zimmer et al. [ZRJ*15] use next-event estimation to find previous correspondences in the path space, but is too costly for real-time applications. Bitterli et al. [MMBJ17] apply a specular reflection motion vector to track the movement of glossy reflections, but is less accurate when the materials are rougher. Hirvonen et al. [HSAS19a] find more accurate virtual image locations reflected from curved surfaces, but still for pure specular reflections only. Our method continues this line of research, but brings out more types of effects and more reliable motion vectors to use in practice.

Filtering for individual effects. The computation of our motion vectors are inspired by various works on individual effects. The percentage closer soft shadows (PCSS) [Fer05] records an average blocker depth and the light size to estimate the size of soft shadows, which gives us insight on tracking ray traced shadows over time. The screen space reflection (SSR) [THS*15] traces glossy reflected objects in the screen space and filter the noisy results. We build upon this idea to compute stochastic glossy reflection motion vectors. The idea of linearly transformed cosines (LTC) method [HDHN16] calculates accurate shading results without the shadows, making it possible for us to consider only indirect illumination in our dual motion vectors for occlusions.

Other real-time schemes utilizing temporal information. Xiao et al. [XNC*20] present a temporal convolutional neural network, taking color, depth, and motion vectors as inputs. It can upsample the highly aliased input imagery to the 4×4 high fidelity and temporally stable results in real-time. Tatzgern et al. [TMKS20] and Lin et al. [LY20] combine spatiotemporal filtering with their many-light sampling methods to achieve real-time global illumination. These methods highly rely on temporal information. So, they can simply replace traditional motion vectors with ours, and they will immediately obtain better temporal consistency and fewer temporal artifacts.

3. Background and Motivation

In this section, we briefly go over the back-projection technique in TAA and RTRT methods, and explain how it is used in temporal filtering.

When two consecutive frames $i-1$ (previous) and i (current) are given, the idea of back-projection is to find for each pixel x_i intersected by the primary ray, where its world-space shading point s_i was in the previous frame at x_{i-1}. As mentioned in Sec. 1, since we know the entire rendering process, the back-projection process can be accurately computed: first, project the pixel x_i back to its world coordinate in the i-th frame, then transform it back to the $(i-1)$-th frame according to the movement of the geometry, and finally project the transformed world coordinate in the $(i-1)$-th frame back to the image space to get x_{i-1}. Denote P as the viewport*modelviewprojection transformation per frame and T as the geometry transformation between frames, the back-projection process can be formally written as

$$x_{i-1} = P_{i-1} T^{-1} P_i^{-1} x_i, \tag{1}$$

where the subscripts represent different frames. According to Eqn. 1, the motion vector $m(x_i) = x_{i-1} - x_i$ is defined as the difference between the back-projected pixel and the current pixel in the image space.

The motion vector for each pixel x_i is computed together with the rendering process, and can be acquired almost without any performance overhead. With the motion vectors, temporal filtering becomes straightforward. In practice, it is a simple linear blending between the current and previous pixel values:

$$\bar{c}_i(x_i) = \alpha \cdot \tilde{c}_i(x_i) + (1-\alpha) \cdot \bar{c}_{i-1}(x_{i-1}), \tag{2}$$

where c is the pixel value, first filtered spatially per frame resulting in $\tilde{c}$, then blended with the pixel value of its previous correspondence $\bar{c}_{i-1}$. The α is a factor between 0 and 1 that determines how much temporal information is trusted and used, usually set to 0.1 to 0.2 in practice, indicating heavy temporal dependence on previous frames. The temporal filtering process continues as more and more frames are rendered, accumulating to a cleaner result. Thus we use the $\bar{}$ and $\tilde{}$ symbols to indicate less and more noise, respectively.

From Eqn. 2, we can see that one frame's contribution over time is an exponential falloff. Thus, if the motion vectors cannot accurately represent correspondence between adjacent frames, ghosting artifacts will appear. Fig. 1 illustrates three typical failure cases of shadows, glossy reflections and occlusions. In these cases, the shadow receivers, reflectors, background appearing from previously occluded regions are all static, resulting in zero-length motion vectors. But these motion vectors are not the desired correspondences between frames. For example, as one would expect intuitively, when a shadow moves, the motion vectors should move along with the shadow, rather than the shadow receiver.

Various methods are designed to alleviate the temporal failure. Salvi et al. [Sal15] proposes to clamp the previous pixel value $\bar{c}_{i-1}(x_{i-1})$ to the neighborhood of the current pixel value, in order to suppress the ghosting artifacts and provide a faster rate of convergence to the current frame. The SVGF method [SKW*17] focuses on a better spatial filtering scheme to acquire $\tilde{c}_i(x_i)$ by considering spatial and temporal variances together. And the A-SVGF method [SPD18] detects rapid temporal changes to adjust the blending factor α to rely more or less on spatial filtering, trading ghosting artifacts for noise.

In contrast to the previous methods, we intend to better utilize the previous information, i.e., we would like to find a more reliable $\bar{c}_{i-1}(x_{i-1})$ so that minimal special treatment is further needed. Our insight is that for shadows and glossy reflections, it is not the geometry in a pixel that we want to track in the previous frame, but the position of the shadow and the reflected virtual image. For occlusions, it is easier for the previously occluded regions in the background to find correspondences also in the background rather than on the occluder.

4. Temporally Reliable Motion Vectors

In this section, we describe our temporally reliable motion vectors. Specifically, we will focus on the three commonly encountered temporal failure cases: shadows, glossy reflections and occlusions.

The high level idea is that we design a separate motion vector for each of these effects. In accordance with the different types of motion vectors, we also treat our final output as a collection of separate

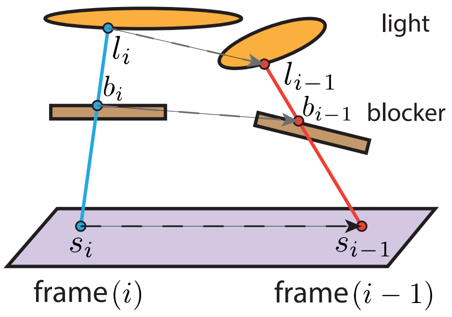

Figure 2: *Illustration of the computation of our shadow motion vectors. We track the movement of the blocker and the light source, and re-calculate the shadow position in the previous frame.*

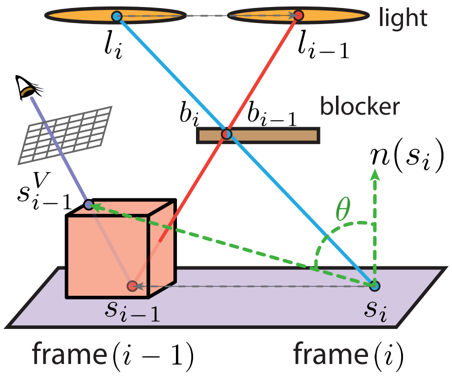

Figure 3: *Illustration of the non-planar shadow receiver issue. It is possible that the back-projected shading point s_{i-1}^V is not on the same virtual receiver plane where s_i resides. Our falloff heuristic detects this case by measuring the extent of "non-planarity", i.e. the angle θ between the normal of the shading point s_i and the direction from s_i to s_{i-1}^V.*

components: shading (direct illumination without shadows), visibility (shadows), glossy reflections (indirect illumination for specular / glossy materials) and indirect illumination for all other materials. The separation is straightforward in an RTRT framework, and it is efficient to combine them for the final result.

4.1. Shadows

Inspired by the Percentage Closer Soft Shadows (PCSS) [Fer05] which estimates the shadow size based on the average blocker depth and light size, we propose to track the movement of the shadow by following the blocker and light positions over time.

Fig. 2 illustrates our scheme focusing on two consecutive frames $(i-1)$ and i. Recall that we shoot 1 shadow ray per pixel towards a randomly chosen position on the light. For a pixel x_i (in image space) in shadow, we know exactly its shading point s_i, the blocker position b_i, and the light sample position l_i (all in world space). Since the blocker and the light sample positions are associated with certain objects, we immediately know their transformation matrices between these two frames, so we are able to find their world space positions b_{i-1} and l_{i-1} in the $(i-1)$-th frame. Suppose the geometry around the shadow receiver s_i is a locally flat plane, we can find the intersection s_{i-1} between this plane and the line connecting l_{i-1} and b_{i-1} in the previous frame. Finally, we project this intersection to the screen space. In this way, this projected pixel x_{i-1}^V is our tracked shadow position from x_i:

$$x_{i-1}^V = P_{i-1}\,\text{intersect}[T^{-1}l_i \to T^{-1}b_i; T^{-1}\text{plane}(s_i)], \quad (3)$$

where the transformations T between frames can be different for the light sample, blocker and the shading point.

Eqn. 3 implies our shadow motion vector as $m^V(x_i) = x_{i-1}^V - x_i$. To use it, we slightly modify the the temporal filtering Eqn. 2 by adding a lightweight clean-up filtering pass (at most 9×9) after temporal blending. This is because the motion vectors $m^V(x_i)$ can be noisy due to random sampling on the light, so the fetched $\bar{V}_{i-1}(x_{i-1})$ can be noisy as well, despite the smoothness of $\bar{V}_{i-1}$ itself in the previous frame. The same clean-up filter will be used for glossy reflections and occlusions, and will be discussed in detail in Sec. 5.

With the clean-up filter, it does not matter whether the shadows are hard or soft. For hard shadows, the blocker and light positions will also be accurate, thus the result will be clean already without any spatial-temporal filtering. For soft shadows, our shadow motion vectors are indeed more noisy because of sampling a large area light, but the penumbra regions will also be spatially filtered more aggressively with larger filter size, thus still resulting in negligible noise. In the accompanying video, we show that our method excels at all different extents of soft shadows.

To perform spatial filtering of the noisy shadows in the current frame i, we refer to Liu et al. [HA19], which accurately calculates the filter size. However, we notice that based on the above computation, only those pixels in shadows in the current frame are associated with our shadow motion vectors. To deal with this problem, and to achieve both efficient filtering performance and clean shadow boundaries, we conceptually interpret the filtering of shadows as the splatting of each in-shadow pixel's visibility, along with other associated properties.

In practice, the splatting is still implemented using a fixed-size filter. Discrete in-shadow pixels' visibilities and their shadow motion vectors are propagated (weight averaged) into the center of each filter. In this way, we are able to determine the accurate shadowed regions. Within the shadows, our shadow motion vectors are always available pixel-wise.

Discussion: multiple lights. Combining with orthogonal approaches, such as Resampling Importance Sampling (RIS) work [BWP*20], our method naturally extends to multiple lights with negligible overhead. The orthogonal method addresses the problem of many-light sampling, and we perform as usual – for each light sample, we compute its motion vector. So, our overhead is always negligible as compared to the direct illumination time, since they scale up together with the number of samples. Fig. 10 shows another view of the *fence* scene with multiple light sources.

Discussion: non-planar shadow receiver. The only assumption we make is that the geometry is locally flat for each shading point during the computation of our shadow motion vectors. However, as Fig. 3 shows, after the back-projection in the $(i-1)$-th frame, it is possible that s^V_{i-1}, the shading point of pixel x^V_{i-1}, is not on the virtual receiver plane defined by s_i and its normal $n(s_i)$ (inverse transformed if the shadow receiver moves over time, omitted here for simplicity). These two shading points may not have the same normals, and could even be on different objects. In this case, it seems that our shadow motion vector could no longer be used.

To deal with the problem introduced because of non-planar shadow receivers, we introduce a simple but effective falloff heuristic. That is, we measure the extent of "non-planarity". Fig. 3 illustrates our idea. Once x^V_{i-1} is calculated, we measure the angle θ between the normal of the virtual receiver plane $n(s_i)$ and the direction $s_i \rightarrow s^V_{i-1}$. Our key observation is that, only when θ is close to 90°, we can fully depend on our shadow motion vector. Otherwise, we should trust more on the spatially filtered result. So, we use θ to adjust the α, replacing it with a specific α^V in Eqn. 2 as

$$\alpha^V = 1 - G(\theta - \frac{\pi}{2}; 0, 0.1) \cdot (1-\alpha), \tag{4}$$

where $G(x;\mu,\sigma)$ is a Gaussian function with it peak value normalized to 1, centered at μ and with a standard deviation of σ. Finally, we achieve high quality, non-lagging shadows, and we compare our results with other methods in Fig. 1 and in Sec. 6. Especially, Fig. 9 shows the *Apples* scene with highly curved apples and cloth on the desk, and a rapidly moving area light on the right. We demonstrate that our shadow motion vectors are still capable of producing visually pleasing results on non-planar shadow receivers.

4.2. Glossy Reflections

Similar to tracking the shadows, we can also track the movement of glossy reflections. This is inspired by Zimmer et al. [ZRJ*15] and Mara et al. [MMBJ17] on mirror / specular reflections. They assume that the virtual image is a real object behind the mirror reflector, and calculate the motion vectors of the virtual image instead of the reflector.

However, when it comes to glossy reflection, this approach is no longer valid. Since we trace 1 secondary ray per pixel x_i, we can only importance sample the glossy lobe to decide where to bounce. And the bounced direction can certainly be different to the specular reflected direction. If we follow exactly their approach, an additional specular reflected ray has to be traced per pixel, which is prohibitively expensive.

Our insight is that no matter if we are using the specular or sampled direction, what we need to do is still to find the corresponding pixel in the previous frame of the secondary hit point h_i. However, since glossy BRDFs model a non-delta distribution, multiple pixels may reflect to the same hit point, forming a finite area in the screen space. This indicates that there will be multiple pixels from the previous frame that correspond to x_i in the current frame. Our stochastic glossy motion vector aims at finding one at a time.

Given that the center of a glossy BRDF lobe is usually the strongest, we always have a valid choice to start with. That is, we first assume that the glossy BRDF degenerates to pure specular

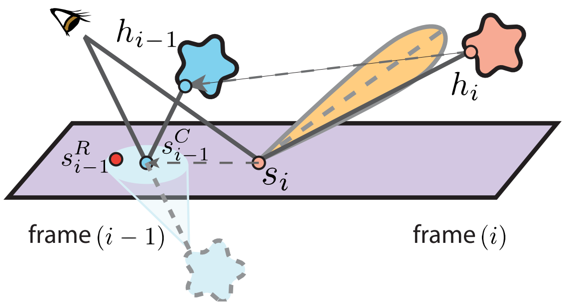

Figure 4: *Illustration of the computation of our stochastic glossy reflection motion vectors. For an importance sampled secondary ray, we back-project the virtual image of its hit point in the previous frame.*

again, then we can immediately find the corresponding point s^C_{i-1} similar to Zimmer et al. [ZRJ*15]. Then our insight is that, as the glossy lobe gradually emerges, a region will appear around s^C_{i-1}, in which all the points are able to reflect to the same hit point h_{i-1}. This region can be approximated by tracing a glossy lobe (with the same roughness at s^C_{i-1}) from the virtual image of h_{i-1} towards s^C_{i-1}.

Fig. 4 illustrates the way we find one corresponding pixel x^R_{i-1} in the previous frame. We start from the importance sampled secondary ray at the shading point s_i and the secondary hit point h_i in the world space. We transform h_i to the previous frame $(i-1)$ in the world space, find its mirror-reflected image, then project it to the screen to retrieve s^C_{i-1} in the world space again.

Then, similar to the shadow case, we assume a locally flat virtual plane around s^C_{i-1}, and find the intersected region between this plane and the glossy lobe traced from the image of h_{i-1} towards s^C_{i-1}. In practice, there is no need to trace any cones, and we simply assume that the glossy lobe is a Gaussian in directions, and that the intersected region is a Gaussian in positions as well as in the image space, which can be efficiently approximated by tracking the endpoints of major and minor axes. In this region, our stochastic motion vector for glossy reflection randomly finds x_i's correspondence at

$$x^R_{i-1} = \text{sample}\left(P_{i-1}\,\text{mirror}[T^{-1}h_i, T^{-1}\text{plane}(s_i)], \Sigma\right), \tag{5}$$

where sample(μ,Σ) is to importance sample a Gaussian function with center μ and covariance Σ, and T still represents different transformations at different places.

The usage of our stochastic glossy reflection motion vectors is similar to the shadow motion vectors. Also, the "natural hierarchy" still exists, i.e., when the roughness is high, the glossy reflection motion vectors will be more noisy, but the temporally filtered result will be then spatially cleaned up in a more aggressive manner.

4.3. Occlusions

Different from the previous cases on shadows and glossy reflection, when occlusion happens, in theory there are no temporal correspondences x^O_{i-1} of the pixels x_i appearing from the previously occluded

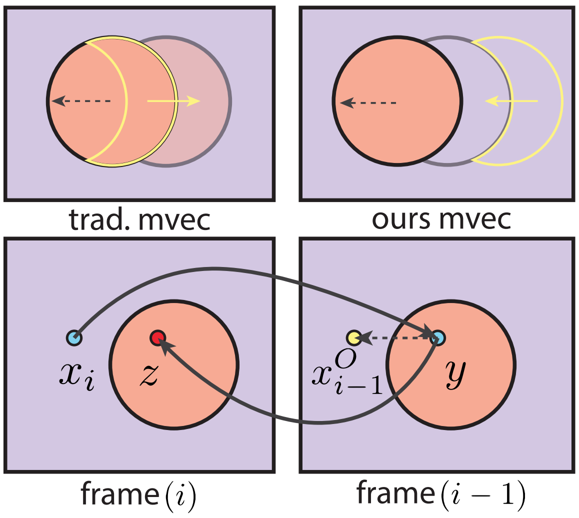

Figure 5: *Illustration of the computation of our dual motion vectors for occlusions, and comparison with traditional motion vectors. For a pixel x_i that is visible now but was occluded in the previous frame at y, we find where the occluder y is in the current frame at z. Then we find x_i's correspondence x^O_{i-1} in the previous frame, assuming that the relative positions between the occluder and the background do not change over time. As for traditional motion vectors (trad. mvec), in this case, its length is always zero. So, it simply reuse the color of the same place in the previous frame. As a result, ghosting artifacts will emerge. On the other hand, our dual motion vectors (ours mvec) predict that temporal information should be acquired from the region in right outlined in yellow.*

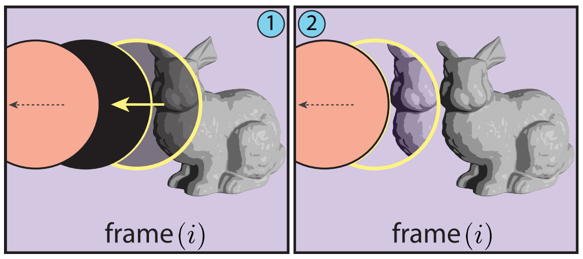

Figure 6: *Illustration of how the repetitive pattern is produced when simply reusing colors with our dual motion vectors. The disk moves from right to left, leaving a previously occluded region (in black) near a stationary bunny. Our dual motion vectors predict that temporal information should be acquired from the region outlined in yellow in the previous frame, so the bunny will simply be repeated.*

regions. The back-projected motion vectors of these pixels will always land on the occluders, thus the previous pixel values cannot easily be used.

To alleviate this issue, we start from the clamping method by Salvi et al. [Sal15]. They evaluate the local neighborhood of x_i in a small window, and fit a Gaussian distribution $G(\mu,\sigma)$ of the color values inside this window. No matter whether the correspondence x^O_{i-1} is valid or not, they clamp the color value at x^O_{i-1} to the $[\mu-\sigma,\mu+\sigma]$ interval, then blend with the color value at x_i using Eqn. 2. This approach improves temporal stability, but is still prone to ghosting artifacts in the occlusion case, because usually the occluders have completely different colors than the unoccluded regions.

Our insight is that if the color value at x^O_{i-1} is closer to the value at x_i, the issues produced by the clamping method can be better resolved. Also, close color values often appear on the same object. Inspired by Bowles et al. [BMS*12], we propose a new motion vector for the just-appeared region to find a similar correspondence in the previous frame. To do that, we refer to the relative motion.

As Fig. 5 shows, the traditional motion vector gives the $x_i \rightarrow y$ correspondence but unfortunately cannot be easily used. Our method continues to track the movement of $y \rightarrow z$ from the previous frame to the current, using the motion of the occluder. Then, based on the relative positions of x_i and z, we are able to find the location x^O_{i-1} in the previous frame. This process can be simply represented as

$$x^O_{i-1} = y + (x_i - z), \tag{6}$$

where $y = P_{i-1}T^{-1}(x_i)P_i^{-1}x_i$ and $z = P_iT(y)P_{i-1}^{-1}y$.

Eqn. 6 indicates that we have applied a back-projection followed by a forward-projection, essentially using two motion vectors. Thus we name our approach "dual motion vectors" for occlusions. In this way, we are able to find a correspondence x^O_{i-1} with a much closer color value to x_i. Note that since we use P of two frames to track x_i, we are able to support the movement of the camera and objects simultaneously.

Note that since we deal with different effects separately for shadows and glossy reflection, and the shading part can already be reasonably approximated in a noise-free way (e.g. using the Linear Transformed Cosines (LTC) method [HDHN16]), we only have to apply our dual motion vectors to indirect illumination. Moreover, since glossy indirect illumination has been elegantly addressed using our glossy reflection motion vectors, we can now focus only on diffuse materials.

Discussion: reusing color vs. incident radiance. However, as Fig. 6 indicates, simply applying the color values as in Bowles et al. [BMS*12] using the dual motion vector will result in clear repetitive pattern, because it is essentially copy-pasting image contents. This is especially problematic when the normals at s_i and s^O_{i-1} are different. And removing the textures from colors (named *demodulation* by the industry) will not help, because when the normals differ, the intensity of shading could already change a lot.

To address this issue, we propose to temporally reuse the incident radiance instead of the shading result. Specifically, for the application of diffuse indirect illumination, we record the 2D indirect incident radiance per pixel. This immediately introduces two questions. First, why using the incident radiance is superior to using the shaded result, especially given that the BRDFs of materials are diffuse. Second, how to efficiently store the 2D incident radiance per pixel, since naïvely storing all incident directions could be prohibitively expensive.

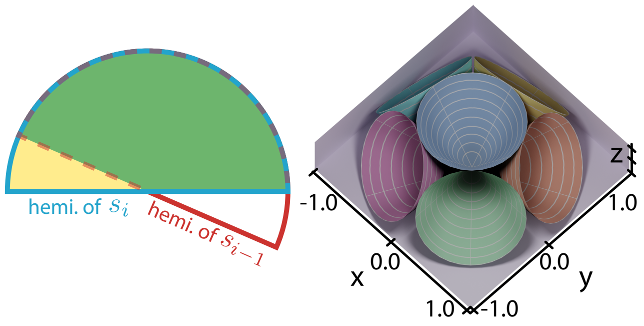

Figure 7: *Left: illustration of our partial temporal reuse scheme. Only the same directions in the overlapped solid angle will share temporally coherent radiance. Right: we use 6 slightly overlapping cones on a hemisphere to store incident radiance approximately.*

For the first question, it is based on the observation that the indirect incident radiance field is usually much smoother than the indirect shading result, because they are irrelevant of the normals at different shading points. As long as two shading points are close in positions, the incident radiance from similar incident directions will be similar as well. This also inspires us that the incident radiance should be recorded and reused in world coordinates. Fig. 7 (left) shows an example. Suppose that we have recorded the incident radiance of s_i and s^O_{i-1}, each on a hemisphere, we immediately know that all the directions in the overlapped regions of these two hemispheres (marked as green) could be temporally reused, while the non-overlapping part (marked as yellow) should remain using only the spatial content from frame i. Then, the radiance from both parts will be used to re-illuminate the shading point x_i, leading to an accurate temporally accumulated shading result.

The use of incident radiance reminds us of the second question. Indeed, we cannot afford storing and blending the incident radiance from every possible direction. But fortunately, efficient representations exists. Ramamoorthi and Hanrahan [RH01] demonstrated that for diffuse materials, it is efficient enough to use only 6 spherical harmonics basis functions to represent the incident radiance. Compact as the representation, it is not as efficient to apply in our case, because we need to quickly figure out a "windowed" spherical function by zeroing out values outside a certain colored region. This involves the triple product operation of spherical harmonics, which was demonstrated extremely inefficient by Ng et al. [NRH04].

With the difficulty of spherical harmonics, we instead refer to the representation in the voxel cone tracing approach [CNS*11]. We subdivide a hemisphere into 6 slightly overlapping cones with equal solid angles $\pi/3$ pointing at different directions, and assume that the radiance remains constant within each cone. During spatial or temporal filtering, instead of averaging the shading result, we filter for each cone individually, using the overlapping solid angles between each pair of cones as the filtering weight. Fig. 7 (right) illustrates our idea. Finally, we are able to achieve a much cleaner result for diffuse indirect illumination, as demonstrated in Fig. 1 and Sec. 6.

5. Implementation Details

Clamping. Our motion vectors are indeed able to find more accurate correspondences of a pixel, however, this does not mean the color of temporally corresponding pixels should be exactly the same. Also, in practice, directly using motion vectors still produces minor artifacts. Therefore, we use a mild clamping operation to limit the temporal color / radiance value x_{i-1} (still found using our motion vectors) to the $[\mu-3\sigma, \mu+3\sigma]$ interval in each x_i's neighborhood to avoid potential artifacts and increase robustness. It is important to note that, this clamping operation does not reintroduce any lagging of shadows, reflections or occlusions like before, but only does the clean-ups, thanks to temporally reliable motion vectors that accurately track the most reliable contents.

The size of the clean-up filter. As mentioned earlier, there are times that we rely more on the spatial filtering. Specifically, one unique case of ours is the change of α according to our non-planar heuristic in shadows. All the other scenarios give credit to clamping, since clamping is essentially transforming the temporal contributions towards the spatial. In either case, we know that the spatially filtered results will be used more, so they cannot be too noisy. For this reason, we determine the size of our clean-up filter based on the value of α and the extent of clamping to further improve the quality of filtering.

For the change of α to α^V, we simply refer to how much it has changed from the default choice ($\alpha = 0.1$ in our case):

$$w_A(x_i) = (\alpha^V(x_i) - \alpha)^2, \tag{7}$$

which measures the extent of how much more we rely on spatial filtering.

For clamping, let's denote $\bar{c}^*_{i-1}(x_{i-1})$ as the clamped history value, then we define the extent of clamping by measuring the relative "lost of information" as

$$w_C(x_i) = \frac{|\bar{c}^*_{i-1}(x_{i-1}) - \bar{c}_{i-1}(x_{i-1})|}{\bar{c}_{i-1}(x_{i-1}) + \varepsilon}, \tag{8}$$

where the ε is set to 0.001 for safe division.

Finally, we use $w_A(x_i)$ and $w_C(x_i)$ together to determine the standard deviation of our clean-up filter as $\min\{w_A(x_i) + w_C(x_i), 5\}$. Note that the maximum size (side length in pixels) of the clean-up filter is always 9×9.

Intersection and contribution of cones. In Sec. 4.3, one of the key technical dependencies is the calculation of the overlapping solid angle between each pair of cones on a unit hemisphere. We refer to the solution by Mazonka et al. [Maz12]. Specifically, for a pair of cones with same apex angle $\frac{\pi}{6}$, the overlapping solid angle Ω can be written as:

$$\Omega = 4 \cdot (\beta - \varphi \cdot \cos\frac{\pi}{6}), \tag{9}$$

where β and φ can be computed as:

$$\begin{aligned} \gamma &= \arctan\frac{\cos\frac{\pi}{6} - \cos\alpha \cdot \cos\frac{\pi}{6}}{\sin\alpha \cdot \cos\frac{\pi}{6}}, \\ \beta &= \arccos\frac{\sin\gamma}{\sin\frac{\pi}{6}}, \\ \varphi &= \arccos\frac{\tan\gamma}{\tan\frac{\pi}{6}}. \end{aligned} \tag{10}$$

where α is the angle between the axes of these two cones.

With the overlapping solid angle Ω between any two cones, we then use it as the weight of contribution during spatial or temporal filtering. After this, we re-shade each pixel using the average radiance recorded in each of the 6 cones, computed as:

$$L \approx f_r \cdot \sum_{k=1}^{6} L_k \cdot \frac{\pi}{3} \tag{11}$$

where f_r is the diffuse BRDF, L_k is the averaged constant radiance in the k-th cone, and $\pi/3$ is the solid angle of each individual cone.

Choice of spatial filters. One benefit from filtering separately for different effects is that we can use small spatial filters for each of them. For shadows, we simply implement a separable 31×31 joint bilateral filter [KCLU07] and it turns out already good enough. For glossy reflections, we refer to the filtering method in [LLK*19]. For global illumination, we use a 4-level à-trous wavelet filter, essentially 32×32. All the clean-up filters are implemented with as a separable joint bilateral filter with a fixed size 9×9 but with potentially different standard deviations.

6. Results and Comparison

We implement our algorithm using the NVIDIA OptiX [PBD*10] as well as the Unreal Engine 4 [San16]. And we compare our method against the state of the art real-time ray tracing techniques over a variety of scenes. All the experiments are conducted on an NVIDIA TITAN RTX with 24GB of video memory. All the images are rendered with 1920×1080 resolution, sampled at 1 light path per pixel, except that the ground truth is traced at 4096 light paths per pixel.

We compare our results with the ones generated using traditional motion vectors, with and without the neighborhood clamping approach used in TAA [Sal15]. The clamping methods represent the line of ideas that force the use of temporal information. We also compare our method with the SVGF and A-SVGF methods as representatives that balance the use of temporal and spatial information. Neither kind of these methods aim at better utilizing the temporal information. We use Root-Mean Square Error (RMSE) and Structural Similarity Index (SSIM) metrics for quantitative evaluation. Note that sometimes SVGF is better in quantity, and this is because it tends to produce a significant amount of overblur. Please make sure to check out our accompanying video for much clearer differences. For the convenience of developers, we will release our full code base upon publication.

Shadows. Fig. 8 (left) shows the *fence* scene with a rapid-moving fence in front and an area light behind it. In this example, we demonstrate that our shadow motion vectors are able to produce shadows that are closely attached to the fence.

In comparison, traditional motion vectors produce significant ghosting artifacts. This is expected, since they will always be zero in this case. With clamping, the results are less lagging but much more noisy. However, the noise is aggressively filtered by SVGF, resulting in overblur. The A-SVGF method discards temporal information, resulting in color blocks similar to the typical "smearing" artifact in bilateral image filtering, and leaving behind low frequency noise that can be easily observed in a video sequence.

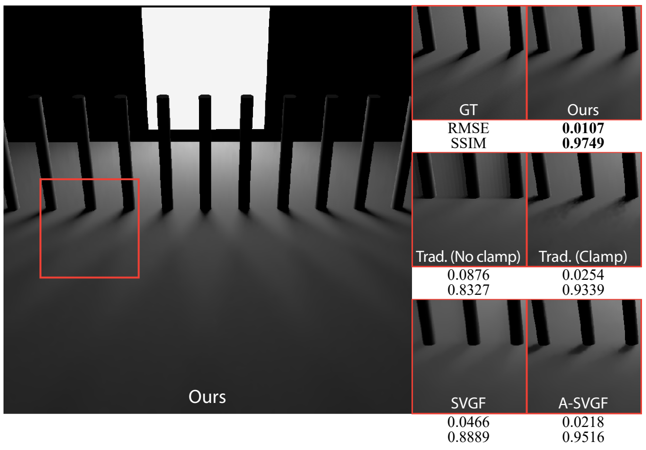

Figure 8: *The* fence *scene with a rapidly moving fence in front and an area light behind it. Our shadow motion vectors are able to produce shadows that are closely attached to the fence.*

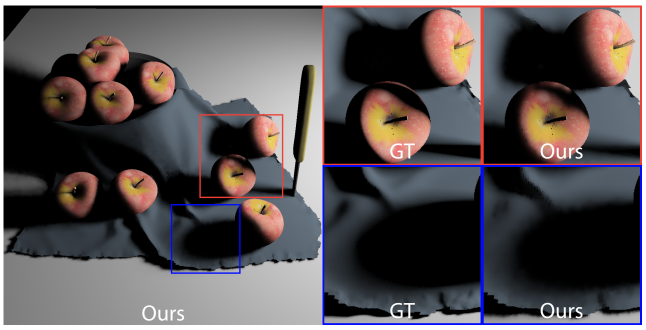

Figure 9: *The* apples *scene with highly curved objects, and a rapidly moving area light on the right. This is to demonstrate that our shadow motion vectors are still capable of producing visually pleasing results on non-planar surfaces.*

We also include two more complex scenes: *pinkroom* scene in Fig. 1 and *apple* scene in Fig. 9. We can see that our shadow motion vectors work well with complex occlusion conditions, even when the locally flat plane assumption is violated, thanks to our cosine falloff heuristic.

Glossy reflections. We show two scenes (Figs. 1 (middle) and 11) that contain glossy reflections of various objects. We compare our method with three approaches: (1) using the traditional reflectors' motion vectors but without clamping of previous pixel values, (2) using traditional motion vectors with clamping, (3) using specular reflected rays' hit points' motion vectors, also with clamping and (4) using traditional motion vectors with the temporal component of A-SVGF (the Temporal Gradient method). For all the comparisons, we use the spatial filter discussed in [LLCK19] as the spatial component of our denoising pipeline.

The comparison indicates that our glossy reflection motion vectors do not introduce ghosting artifacts. However, with traditional motion vectors, naive filtering produces significant ghosting.

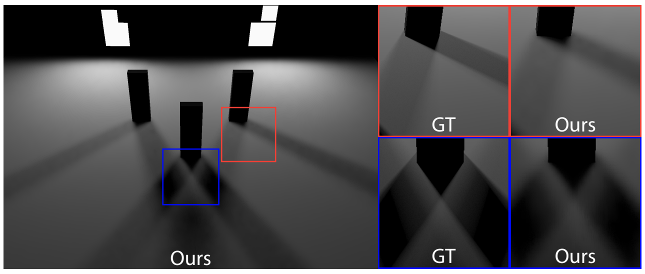

Figure 10: *Another view of the* fence *scene with multiple moving light sources. Compared with the ground truth, we found that our result tends to be blurred and less plausible.*

Clamping relieves the lagging but introduces severe discontinuous artifacts. With specular motion vectors, the results look plausible in most regions, but discontinuous artifacts can still be found around the edges of the reflected objects. The A-SVGF will always result in noisy results since the temporal gradient changes so fast that mostly the A-SVGF only uses the noisy current frame. Our method produces the closest result to the ground truth (ray traced without firefly removal thus always looks brighter).

Occlusions. We demonstrate the effectiveness of our occlusion motion vectors in two different versions of the *PICA* scene with moving objects in Figs. 1 (right) and 12. Only indirect illumination is shown and its intensity is scaled 10× for better visibility. Note that this will also make the artifacts and low-frequency noises more visible than usual. New unoccluded regions will appear around the boundaries of foreground objects. In these regions, the temporal information will simply be rejected with traditional motion vectors. Therefore, in this case the SVGF still results in significant amount of overblur, while A-SVGF again appears to be smeared spatially and loses temporal stability. The clamping approach tries to use pixel values from the occluders, however, since the pixel values on the foreground and background usually differ drastically in the occlusion case, this will still introduce ghosting artifacts.

7. Discussion and Limitation

Potential improvements with orthogonal approaches. Since our method focuses on reliable temporal information, any orthogonal method that improves spatial filtering is potentially helpful to our method. These include exploiting blue noise sampling patterns [HB19], better shadow estimators [HHM18], block-wise filtering [KIM*19], and more. Also, there are better methods that help us to find more accurate motion vectors, for example, better ways of finding the mirror-reflected images by curved reflectors [HA19]. Moreover, adaptive methods that dynamically select temporal blending weights such as A-SVGF [SPD18] can also be combined with our method immediately for better temporal robustness.

Performance. Fig. 13 shows average computation cost of each step of denoising individual effect using our motion vectors. The average cost of individual step is estimated from 500 frames rendered at 1920 × 1080 on an NVIDIA TITAN RTX. We compared with SVGF (traditional motion vectors), it can denoise different effects with a similar cost, around 3.11 ms per frame.

Different from previous work, our main contribution is not a spatial filtering approach or system, but the different types of motion vectors. As one would expect from their simple computation in Sec. 4, in practice, we observed only a negligible performance cost by replacing with our motion vectors, which is always less than 0.23 ms. Besides, we have also noticed that our implementation of denoising shadows and glossy reflections is already much faster than SVGF, since we use much simpler spatial filters and fewer levels or passes (Sec. 5). Finally, when denoising indirect illumination, it is worth mentioning the additional cost when we introduce the cones for storing and filtering the incident radiance. For this, compared with reusing colors, we found that the typical cost of denoising each frame increased around 1.5 milliseconds.

Note that our total computation cost for denoising all effects is higher than SVGF. This is the drawback and inevitable expense of separably filtering. However, the industry has been claiming and in practice demonstrating that separably filtering each effect is overall better and preferred [Liu18].

Glossy reflections on the curved surfaces. Fig. 14 shows our result on highly curved surfaces. Compared with results on planar surfaces, it is less plausible. This is because we did not specially design our stochastic motion vector for finding the mirror-reflected image on curved surfaces. However, there is a practically good way of doing this using the thin lens approximation [HSAS19b]. Again, this orthogonal method will only change the way we find the virtual image. Our stochastic motion vector part remains unchanged with or without it.

Ghosting due to the sudden change of local irradiance. By temporally reusing the incident radiance, we address the ghosting issue shown in Fig. 6. However, sometimes our method will still introduce faintly visible ghosting due to the intensely varied irradiance in the local area. As shown in Fig. 15 (a), the foreground box move from left to right, continuously bringing high irradiance to low brightness regions, and leaving the artifact along the way. Besides, this ghosting will be more noticeable without our clean-up filters. On the other hand, without the 10 × brightness scale of indirect illumination, this ghosting is acceptable in most scenarios.

Overblur of shadows. As shown in Fig. 15 (b), the shadow of the vase is overly blurred. This is because our method will also produce overblur of shadows, especially for thin and small shadows. Although our shadow motion is tailored for temporally filtering shadows, it can not address the ubiquitous overblur of existing shadow filtering methods. Meanwhile, any orthogonal state-of-the-art filter may help us improve the final quality of shadows in the future.

The out of image plane issue. Currently, we do not explicitly handle the cases where the motion vectors point to regions outside the image plane. However, as shown in Fig. 15 (c), this is a commonly encountered scenario, especially when the camera is moving and new areas come inside the current image plane. Similar image space issues may also arise when the shadows or reflections move behind foreground objects. In these cases, it might be possible to extend our dual vectors approach to find existing regions to replace

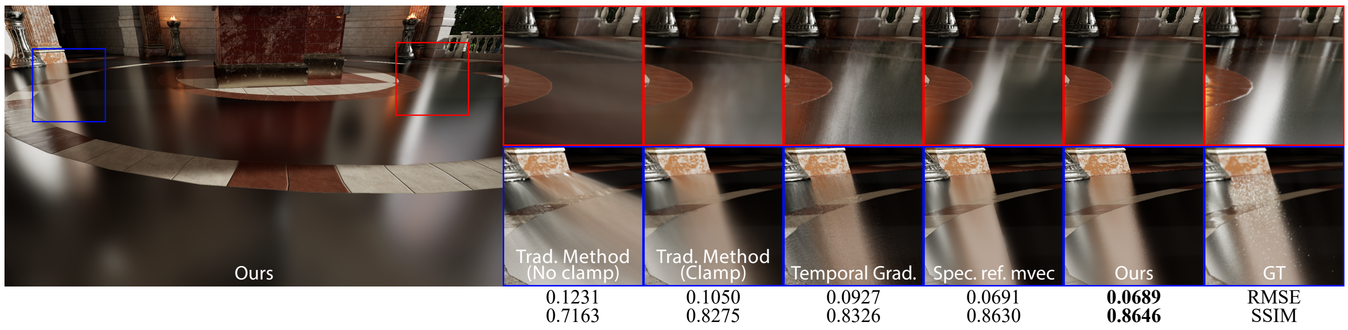

Figure 11: *The* sun temple *scene with a rapidly moving camera. Our stochastic glossy reflection motion vector is able to produce accurate reflections, while the traditional motion vectors result in significant ghosting artifacts. The clamping and A-SVGF (temporal gradient) reduce the lagging, but introduce discontinuous artifacts and noise respectively. Using specular reflection motion vectors greatly relieves the ghosting, but produces unrealistic boundaries on the reflected objects.*

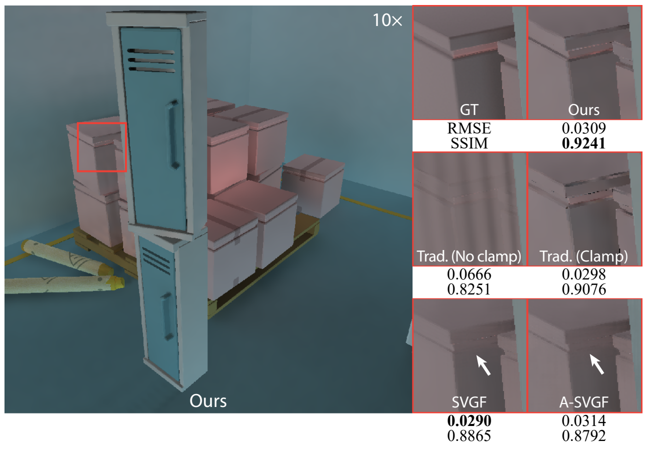

Figure 12: *Another view of the* PICA *scene with objects moving from left to right. For these regions appearing from previous occlusions, our occlusion motion vectors are able to find their corresponding locations in the previous frame, thus avoiding noise and repetitive patterns due to invalid motion vectors, preserving details and suppressing overblur, color blocks, and temporal instability compared to SVGF/A-SVGF.*

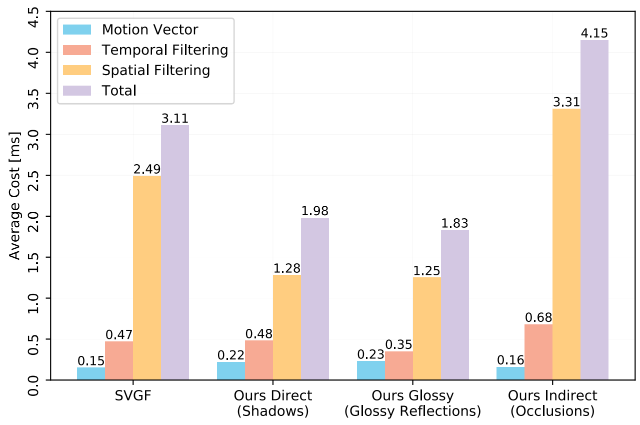

Figure 13: *Runtime breakdown of our methods. See Sec. 7 for details.*

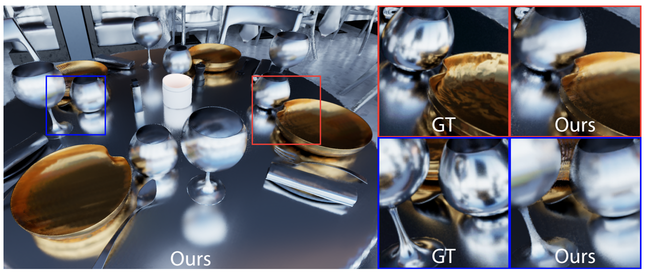

Figure 14: *The* contemporary restaurant *scene with highly curved objects and a rapidly moving camera. See Sec. 7 for details.*

the regions outside, but for now we simply rely on a larger spatial filter for these regions.

Failure due to drastic temporal change. There are cases when the temporal change is too drastic so there is barely any temporal information that can be used. One typical case is the sudden switch of scenes, where all temporal information is invalid. Also, in extreme cases, for example, thousands of leaves are covering the entire image plane but suddenly some background leaks out, any motion vector will not be able to find regions similar to the background. However, our method still would not behave worse than traditional motion vectors in these extreme cases.

Failure due to mixed information. Currently, we only record one motion vector per pixel. However, in certain cases, it is possible that one pixel can be associated with multiple motion vectors. For example, as shown in Fig. 15 (d), when multiple occluders cast shadows onto the same region, or when the BRDF of a shading point contains multiple glossy lobes, recording only one motion vector per pixel is not enough. Another example is in Fig. 10. It shows another view of the *fence* scene with multiple light sources.

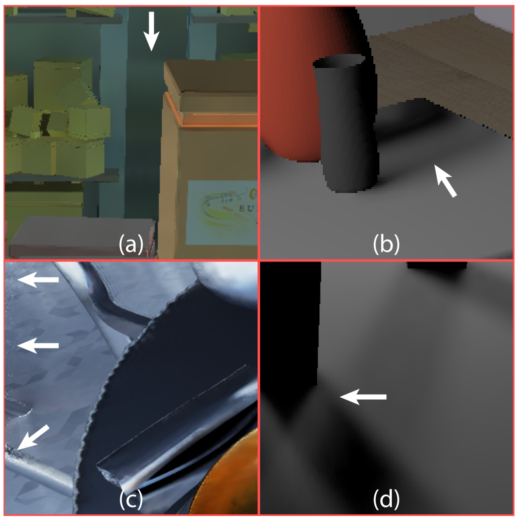

Figure 15: *Failure cases of our methods. (a) Ghosting due to the sudden change of local irradiance. (b) Overblur of shadows. (c) The out of image plane issue. (d) Failure due to mixed information. See Sec. 7 for details.*

Compared with the ground truth, we found that our result tends to be blurred and less plausible.

8. Conclusion and Future Work

We have proposed multiple types of motion vectors for better utilization of temporal information in real-time ray tracing. With our motion vectors, we are able to track the movement of shadows and glossy reflections, and find similar regions to blend with previously occluded regions. We show that our motion vectors are temporally more reliable than traditional motion vectors, and show cleaner results compared to the state of the art methods with negligible performance overhead. We also demonstrate that our method works in both experimental and commercial real-time ray tracing frameworks, and is production ready.

In the future, we would like to design temporally reliable motion vector for distribution effects, such as motion blur and depth of field effects. It would also be interesting to keep multiple previous frames to study non-exponential temporal falloffs. Exploiting machine learning approaches to better summarize temporal correlations from examples could also be a promising direction.

Acknowledgement. This work has been partially supported by the National Key R&D Program of China under grant No. 2020YFB1709200 and the National Natural Science Foundation of China under grant No.61872223.

References

[BMS*12] Bowles, Huw, Mitchell, Kenny, Sumner, Robert W, et al. "Iterative image warping". *Computer graphics forum*. Vol. 31. 2pt1. Wiley Online Library. 2012, 237–246 6.

[BRM*16] Bitterli, Benedikt, Rousselle, Fabrice, Moon, Bochang, et al. "Nonlinearly Weighted First-order Regression for Denoising Monte Carlo Renderings". *Computer Graphics Forum*. Vol. 35. 4. Wiley Online Library. 2016, 107–117 2.

[BWP*20] Bitterli, Benedikt, Wyman, Chris, Pharr, Matt, et al. "Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting". *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 2020). DOI: 10/gg8xc7 4.

[CKS*17] Chaitanya, Chakravarty R Alla, Kaplanyan, Anton S, Schied, Christoph, et al. "Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder". *ACM Transactions on Graphics (TOG)* 36.4 (2017), 98 2.

[CNS*11] Crassin, Cyril, Neyret, Fabrice, Sainz, Miguel, et al. "Interactive indirect illumination using voxel cone tracing". *Computer Graphics Forum*. Vol. 30. 7. Wiley Online Library. 2011, 1921–1930 7.

[EHDR11] Egan, Kevin, Hecht, Florian, Durand, Frédo, and Ramamoorthi, Ravi. "Frequency Analysis and Sheared Filtering for Shadow Light Fields of Complex Occluders". *ACM Transactions on Graphics* 30.2 (Apr. 2011), 1–13. URL: http://graphics.berkeley.edu/papers/Egan-FAS-2011-04/ 2.

[Fer05] Fernando, Randima. "Percentage-closer soft shadows." *SIGGRAPH Sketches*. 2005, 35 3, 4.

[HA19] Haines, Eric and Akenine-Moller, Tomas. "Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs". (2019) 4, 9.

[HB19] Heitz, Eric and Belcour, Laurent. "Distributing Monte Carlo Errors as a Blue Noise in Screen Space by Permuting Pixel Seeds Between Frames". *Computer Graphics Forum*. Vol. 38. 4. Wiley Online Library. 2019, 149–158 9.

[HDHN16] Heitz, Eric, Dupuy, Jonathan, Hill, Stephen, and Neubelt, David. "Real-time polygonal-light shading with linearly transformed cosines". *ACM Transactions on Graphics (TOG)* 35.4 (2016), 41 3, 6.

[HHM18] Heitz, Eric, Hill, Stephen, and McGuire, Morgan. "Combining analytic direct illumination and stochastic shadows". *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM. 2018, 2 9.

[HS81] Horn, Berthold KP and Schunck, Brian G. "Determining optical flow". *Artificial intelligence* 17.1-3 (1981), 185–203 2.

[HSAS19a] Hirvonen, Antti, Seppälä, Atte, Aizenshtein, Maksim, and Smal, Niklas. "Accurate Real-Time Specular Reflections with Radiance Caching". *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*. Ed. by Haines, Eric and Akenine-Möller, Tomas. Berkeley, CA: Apress, 2019, 571–607. ISBN: 978-1-4842-4427-2. DOI: 10.1007/978-1-4842-4427-2_32. URL: https://doi.org/10.1007/978-1-4842-4427-2_32 3.

[HSAS19b] Hirvonen, Antti, Seppälä, Atte, Aizenshtein, Maksim, and Smal, Niklas. "Accurate Real-Time Specular Reflections with Radiance Caching". *Ray Tracing Gems*. Springer, 2019, 571–607 9.

[IMK*16] Iglesias-Guitian, Jose A, Moon, Bochang, Koniaris, Charalampos, et al. "Pixel History Linear Models for Real-Time Temporal Filtering". *Computer Graphics Forum*. Vol. 35. 7. Wiley Online Library. 2016, 363–372 2.

[KCLU07] Kopf, Johannes, Cohen, Michael F, Lischinski, Dani, and Uyttendaele, Matt. "Joint bilateral upsampling". *ACM Transactions on Graphics (ToG)* 26.3 (2007), 96–es 8.

[KIM*19] Koskela, Matias, Immonen, Kalle, Mäkitalo, Markku, et al. "Blockwise Multi-Order Feature Regression for Real-Time Path Tracing Reconstruction". *ACM Transactions on Graphics (TOG)* 38.5 (June 2019). DOI: 10.1145/3269978 2, 9.

[KTD*14] Karis, Brian, Tatarchuk, Natasha, Drobot, Michal, et al. "Advances in real-time rendering in games, part I". *ACM SIGGRAPH 2014 Courses*. ACM. 2014, 10 2.

[Liu18] LIU, EDWARD. *Low Sample Count Ray Tracing with NVIDIA's Ray Tracing Denoisers*. 2018. URL: https://on-demand.gputechconf.com/siggraph/2018/video/sig1813-5-edward-liu-low-sample-count-ray-tracing-denoisers.html 9.

[LLCK19] LIU, EDWARD, LLAMAS, IGNACIO, CAÑADA, JUAN, and KELLY, PATRICK. "Cinematic Rendering in UE4 with Real-Time Ray Tracing and Denoising". *Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs*. Ed. by HAINES, ERIC and AKENINE-MÖLLER, TOMAS. Apress, 2019, 289–319 8.

[LLK*19] LIU, EDWARD, LLAMAS, IGNACIO, KELLY, PATRICK, et al. "Cinematic rendering in UE4 with real-time ray tracing and denoising". *Ray Tracing Gems*. Springer, 2019, 289–319 8.

[LY20] LIN, DAQI and YUKSEL, CEM. "Real-Time Stochastic Lightcuts". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3.1 (2020), 1–18 3.

[Maz12] MAZONKA, OLEG. "Solid angle of conical surfaces, polyhedral cones, and intersecting spherical caps". *arXiv preprint arXiv:1205.1396* (2012) 7.

[MHC*16] MUNKBERG, JACOB, HASSELGREN, JON, CLARBERG, PETRIK, et al. "Texture space caching and reconstruction for ray tracing". *ACM Transactions on Graphics (TOG)* 35.6 (2016), 249 2.

[MIYM15] MOON, BOCHANG, IGLESIAS-GUITIAN, JOSE A, YOON, SUNG-EUI, and MITCHELL, KENNY. "Adaptive rendering with linear predictions". *ACM Transactions on Graphics (TOG)* 34.4 (2015), 121 2.

[MMBJ17] MARA, MICHAEL, MCGUIRE, MORGAN, BITTERLI, BENEDIKT, and JAROSZ, WOJCIECH. "An efficient denoising algorithm for global illumination." *High Performance Graphics*. 2017, 3–1 2, 3, 5.

[MMMG16] MOON, BOCHANG, MCDONAGH, STEVEN, MITCHELL, KENNY, and GROSS, MARKUS. "Adaptive polynomial rendering". *ACM Transactions on Graphics (TOG)* 35.4 (2016), 40 2.

[MWR12] MEHTA, SOHAM UDAY, WANG, BRANDON, and RAMAMOORTHI, RAVI. "Axis-aligned filtering for interactive sampled soft shadows". *ACM Transactions on Graphics (TOG)* 31.6 (2012), 163 2.

[NRH04] NG, R, RAMAMOORTHI, R, and HANRAHAN, P. "Wavelet triple product integrals for all-frequency relighting". *Acm Transactions on Graphics* 23 (2004), 477–487 7.

[NSL*07] NEHAB, DIEGO, SANDER, PEDRO V, LAWRENCE, JASON, et al. "Accelerating real-time shading with reverse reprojection caching". *Graphics hardware*. Vol. 41. 2007, 61–62 2.

[PBD*10] PARKER, STEVEN G, BIGLER, JAMES, DIETRICH, ANDREAS, et al. "OptiX: a general purpose ray tracing engine". *Acm transactions on graphics (tog)* 29.4 (2010), 1–13 8.

[PSK*16] PATNEY, ANJUL, SALVI, MARCO, KIM, JOOHWAN, et al. "Towards foveated rendering for gaze-tracked virtual reality". *ACM Transactions on Graphics (TOG)* 35.6 (2016), 179 2.

[RH01] RAMAMOORTHI, RAVI and HANRAHAN, PAT. "On the relationship between radiance and irradiance: determining the illumination from images of a convex Lambertian object". *JOSA A* 18.10 (2001), 2448–2459 7.

[Sal15] SALVI, M. "Anti-Aliasing: Are We There yet". *Open Problems in Real-Time Rendering, SIGGRAPH Courses* (2015) 3, 6, 8.

[San16] SANDERS, ANDREW. *An Introduction to Unreal Engine 4*. AK Peters/CRC Press, 2016 8.

[SKW*17] SCHIED, CHRISTOPH, KAPLANYAN, ANTON, WYMAN, CHRIS, et al. "Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination". *Proceedings of High Performance Graphics*. ACM. 2017, 2 2, 3.

[SPD18] SCHIED, CHRISTOPH, PETERS, CHRISTOPH, and DACHSBACHER, CARSTEN. "Gradient estimation for real-time adaptive temporal filtering". *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.2 (2018), 24 2, 3, 9.

[THS*15] TATARCHUK, NATALYA, HILLAIRE, SÉBASTIEN, STACHOWIAK, TOMASZ, et al. "Advances in real-time rendering in games, part I". *ACM SIGGRAPH 2015 Courses*. ACM. 2015, 10 3.

[TMKS20] TATZGERN, WOLFGANG, MAYR, BENEDIKT, KERBL, BERNHARD, and STEINBERGER, MARKUS. "Stochastic Substitute Trees for Real-Time Global Illumination". *Symposium on Interactive 3D Graphics and Games*. 2020, 1–9 3.

[VMCS15] VAIDYANATHAN, KARTHIK, MUNKBERG, JACOB, CLARBERG, PETRIK, and SALVI, MARCO. "Layered light field reconstruction for defocus blur". *ACM Transactions on Graphics (TOG)* 34.2 (2015), 23 2.

[WDP99] WALTER, BRUCE, DRETTAKIS, GEORGE, and PARKER, STEVEN. "Interactive rendering using the render cache". *Rendering techniques' 99*. Springer, 1999, 19–30 2.

[XNC*20] XIAO, LEI, NOURI, SALAH, CHAPMAN, MATT, et al. "Neural supersampling for real-time rendering". *ACM Transactions on Graphics (TOG)* 39.4 (2020), 142–1 3.

[YMRD15] YAN, LING-QI, MEHTA, SOHAM UDAY, RAMAMOORTHI, RAVI, and DURAND, FREDO. "Fast 4D Sheared Filtering for Interactive Rendering of Distribution Effects". *ACM Transactions on Graphics* 35.1 (2015), 7 2.

[ZRJ*15] ZIMMER, HENNING, ROUSSELLE, FABRICE, JAKOB, WENZEL, et al. "Path-space Motion Estimation and Decomposition for Robust Animation Filtering". *Computer Graphics Forum*. Vol. 34. 4. Wiley Online Library. 2015, 131–142 3, 5.