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

Real-time Reflective and Refractive Novel-view Synthesis

Gerrit Lochmann[1] Bernhard Reinert[2] Tobias Ritschel[2,3] Stefan Müller[1] Hans-Peter Seidel[2]

[1] University of Koblenz-Landau [2] MPI Informatik [3] Saarland University

Figure 1: *Novel-view images containing planar and curved reflections and refractions that are correctly warped with our algorithm in real-time. Bottom left: The visual flow between the initial- and the novel-view on the diffuse layer.*

Abstract

We extend novel-view image synthesis from the common diffuse and opaque image formation model to the reflective and refractive case. Our approach uses a ray tree of RGBZ images, where each node contains one RGB light path which is to be warped differently depending on the depth Z and the type of path. Core of our approach are two efficient procedures for reflective and refractive warping. Different from the diffuse and opaque case, no simple direct solution exists for general geometry. Instead, a per-pixel optimization in combination with informed initial guesses warps an HD image with reflections and refractions in 18 ms on a current mobile GPU. The key application is latency avoidance in remote rendering in particular for head-mounted displays. Other applications are single-pass stereo or multi-view, motion blur and depth-of-field rendering as well as their combinations.

1. introduction

Streaming image sequences from a computationally powerful rendering server to a client device such as a mobile phone (client-side rendering) is becoming increasingly popular because it both saves energy and physical size as well as it avoids disclosing content to the client device. The most promising approaches use the client device to merely upsample a low-bandwidth RGBZ image stream with high latency to high temporal and spatial resolution with low latency. In particular when head-mounted displays (HMDs) are used to display the upsampled images, a high temporal resolution is crucial to avoid motion sickness. Commonly depth information is used to warp the image when producing novel frames. Depth-based warping however, does not extend to non-diffuse light transport for two reasons: First, there is no single depth per-pixel and second, multiple layers all warp in

a different way. Second, the warp for curved surfaces is not a linear function of depth. In this paper we show how an RGBZ image ray tree can be used to produce correct multi-layer specular warping, including curved and detailed surfaces.

2. Previous Work

Starting from image warping [Wol98], Chen and Williams [CW93] have shown how synthesis of novel views from rendered images can be used to approximate solutions to several important problems, such as stereo or shadows. Using multiple images, increasing the temporal resolution is a special form of view synthesis [MMB97]. Most view synthesis approaches implicitly assumes that hit points that are warped fall onto a straight line, which is naturally the case for diffuse opaque surfaces and while it still holds for flat mirrors, curved mirrors and refractions behave differently. Layered depth images [SGHS98] (LDIs) help with disocclusions and sampling issues but do not address transparency or help with the limitation of diffuse view synthesis. Non-diffuse view synthesis was first addressed by Lischinski and Rappoport [LR98], who use a cube of LDIs. Their method is best-suited for single-bounce, glossy specular transport, as it involves resampling the light field at every indirection. Earlier work based on re-projection and re-sampling [AH95, BJ88, Che97] does not distinguish between specular and diffuse images. They will work as long as every pixel has one dominant appearance at a single depth on a planar surface but break down if the appearance of a pixel is not dominated by a single depth on a planar surface.

The requirement of ever-increasing spatial and temporal resolution has led to the suggestion to use dedicated hardware that performs only view synthesis [PEL*00]. Our system would agree with such a model, providing its extension to multiple layers and a more involved warping model to account for the non-linearity of curved specular surfaces. An efficient warping to temporally up-sample rendered image streams of disuse opaque surfaces was proposed by Andreev et al. [And10] and Didyk et al. [DER*10]. Even higher performance is achieved, when inverting the flow field, allowing to use gathering-type computation instead of a scattering-type mapping [YTS*11, BMS*12]. An overview of recent advances in remote rendering is given in [SYM*11].

The requirement to split the image into a ray tree has been acknowledged and remains a challenging problem for captured imagery [SKG*12]. Dealing with rendered images, this challenge is not present. Instead, we seek to reduce the computation required and bandwidth used to include specular and transparent effects into remote rendering.

In classic rendering, all information is available to the client, while in classic remote rendering only an RGB image is streamed. Many variants of the in-between continuum exist. One option, is to transmit only the missing pixels by predicting what the client can extrapolate [Lev95] or the differential 3D information required to render the next frame [COMF99].

The advent of consumer-level remote rendering as well as stereo rendering, have renewed the interest in view synthesis. Didyk et al. [DRE*10] have used a stream of high-resolution RGBZ images to produce a stereo image pair with reduced occlusion artifacts by reusing information over time. Remote rendering using consumer GPUs can achieve high performance and good visual quality with low bandwidth, if a diffuse and opaque scene is assumed [PHE*11, PHM*14]. Our specular extension is orthogonal to such approaches: We transmit a full dense depth buffer, but even lower bandwidth would be achieved when transmitting sparse information.

3. Our approach

3.1. Overview

Our system is modeled as client-server architecture (Fig. 2). The server is assumed to be sufficiently powerful to render diffuse global illumination in combination with sharp, non-glossy reflections and refractions. The resulting images are encoded as an RGBZ ray tree(Sec. 3.2). This ray tree is transmitted to the client at a temporal frequency, that is too low to be used for display directly, and it comes with varying delay, potentially dropping frames. The effects of this delay are particularly noticable if the images are displayed on a head-mounted display where latency is to be avoided to prevent motion sickness.

The client is assumed to be able to do simple hardware-accelerated image filtering. It has to produce a high frame-rate and high-resolution image, possibly including stereo view synthesis, depth-of-field, motion or combinations thereof. The client decodes the RGBZ ray tree (Sec. 3.3) and uses image warping of the RGB image in every node based on the Z depth information to produce novel views (Sec. 3.4). The warped images of the ray tree in the novel view are combined, weighted by the view-dependent Fresnel term, to produce the final image. To perform image warping, the forward flow field – a mapping from every pixel location to its location the new image frame – is required. While this flow is routinely computed in closed form for the diffuse case, there is no obvious generalization to reflections on planar or curved surfaces. Refractions, which additionally depend on the index of refraction (IOR) are even more intricate. Efficiently producing this correct flow including the reflective and refractive case is our main contribution.

3.2. Encoding

The images rendered by the server are encoded as an RGBZ ray tree [Whi80] before transmission. In an RGBZ ray tree, every node holds the RGB radiance contribution to the sensor as an image with an additional Z depth channel. Every node contains an image for a specific combination of reflections and refractions of rays that started at the sensor while the root node contains the diffuse color. We add a Z depth component to each such image to allow the client to restore the pixels'

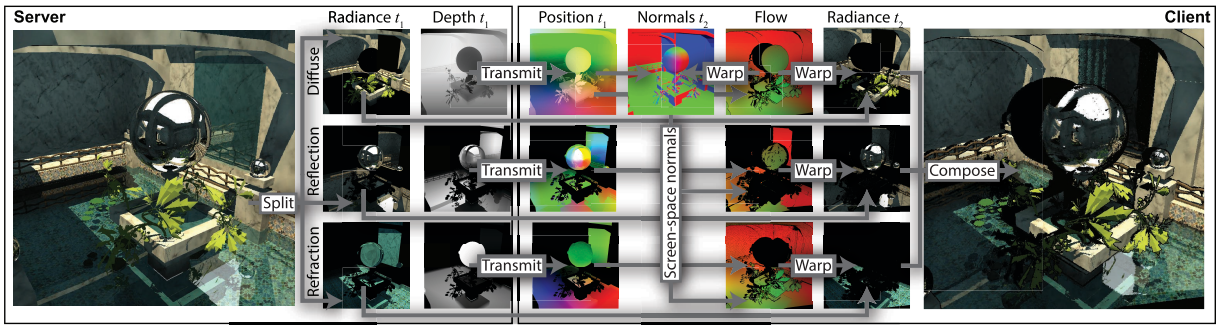

Figure 2: *The server splits the rendering into a diffuse, a reflective and a refractive RGBZ image which are transmitted to the client. To create novel views, the client first decodes the diffuse positions and normals and subsequently the specular positions. These are used to create the flow to be applied to each node. Finally, the warped images are composed for the result image.*

world positions. The ray tree is optionally compressed using an arbitrary RGBZ image compression on each node before transmission.

Radiance The radiance contribution is pre-multiplied by all view-independent factors but not by the view-dependent Fresnel term. The view-independent reflective and refractive "filter colors", such as the green color that an artist has selected to "fake" scattering inside the pool in Fig. 1 are simply multiplied. The view-dependent Fresnel term, however, that controls the relative weight of reflection and refraction is not pre-multiplied, as the weights change drastically for different views at grazing angles which would become noticeable when pre-multiplied. The HDR RGB radiance of each node is tone-mapped before transmitting it using a simple linear tone-mapper. The tonemapper settings are chosen the same for all nodes to deliver the best result for the sum of all nodes i. e., the result image.

Depth Every pixel in every node image has accurate depth information, where depth is given in world space relative to the parent node in the ray tree. The depth of the root node is the distance to the camera, i. e., an ordinary depth image.

Compression Compression is optional and only important if the transmission bandwidth is limited such as over a network. The particular method for compressing the radiance and depth stream is fully orthogonal to our solution. We report results for uncompressed streams, both in RGB and depth. Using a method specifically tailored to encode depth, would further increase transmission performance [PHE*11, PHM*14], orthogonal to the specular warping proposed.

3.3. Decoding

At the client side, the compressed RGBZ ray tree is first decompressed into an uncompressed RGBZ ray tree and subsequently *decoded* into a collection of RGB and XYZ images similar to deferred shading buffers [DWS*88] to simplify and accelerate the following steps. Every node of the ray tree is decoded independently but in pre-order, as child nodes rely on information of their parent nodes (position and normal). The decoding is done independently for each pixel at location $\mathbf{l} \in \mathbb{R}^2 = (x,y)^\mathsf{T}$. The transmitted image is referred to as *old* while the one we seek to extrapolate is called *new*.

Inputs to the decoding are the old model-view-projection matrix $\mathsf{M}^{\text{old}} \in \mathbb{R}^{4\times4}$ used to create the old diffuse image I_d^{old}, the reflective image I_r^{old}, the refractive image I_f^{old} and all images of deeper tree nodes. The camera position $\mathbf{c}^{\text{old}} \in \mathbb{R}^3$ in world space can be found by inversion of the model-view matrix.

Diffuse position A pixel in the old diffuse image is given as $I_\text{d}^{\text{old}}(\mathbf{l}) \in \mathbb{R}^2 \to \mathbb{R}^4 = (r,g,b,z)^\mathsf{T}$ where $(r,g,b)^\mathsf{T}$ defines the pixel color and z gives the depth to the camera. The 3D world position $\mathbf{w}_\text{d} \in \mathbb{R}^3$ at location $\mathbf{l}$ can now be calculated as $\mathbf{w}_\text{d} = \texttt{unproject}(\mathsf{M}^{\text{old}}, (x,y,z)^\mathsf{T})$, where `unproject` is the OpenGL unprojection function. The diffuse world positions at each location are combined in an image W_d.

Normals To compute the specular flow, we need to know the normal for every pixel. Instead of transmitting them as an additional image, we reconstruct them in screen space using the decoded diffuse world positions. At pixel location $\mathbf{l}$ the normal $\mathbf{n}_\text{d} \in \mathbb{R}^3$ is given as the cross product of the connections to two neighboring pixels, i. e.,

$$\mathbf{n}_\text{d} = \Big(W_\text{d}(\mathbf{l}+(0,1)^\mathsf{T}) - W_\text{d}(\mathbf{l})\Big) \times \Big(W_\text{d}(\mathbf{l}+(1,0)^\mathsf{T}) - W_\text{d}(\mathbf{l})\Big).$$

The reconstructed normals exhibit discretization and quantization errors, which are resolved using a 7×7 gaussian blur kernel (Fig. 3). To avoid blurring over edges we experimented with a bilateral filter using the depth values as guidance but opted for the less expensive gaussian filter, which is seperable and can be represented by a mipmap, as it did not cause visible aritfacts in the final images (Fig. 9).

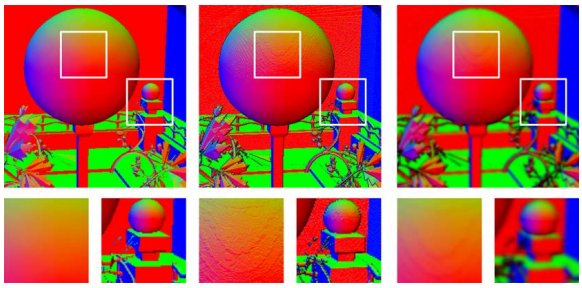

Figure 3: Left: *ground truth.* Center*: reconstructed normals from depth.* Right*: smoothed reconstruction.*

Reflected position Given the diffuse world position of the reflective surface $\mathbf{w}_\mathrm{d}$ at position $\mathbf{l}$, the surface normal $\mathbf{n}_\mathrm{d}$ at this position and the camera position $\mathbf{c}^\mathrm{old}$, we can calculate the transformed (reflected) viewing ray $\mathbf{r} \in \mathbb{R}^3$ at location $\mathbf{l}$ as $\mathbf{r} = \texttt{reflect}(\mathbf{w}_\mathrm{d} - \mathbf{c}^\mathrm{old}, \mathbf{n}_\mathrm{d})$ where `reflect` is the OpenGL reflection function. Given the distance of the pixel to the reflective surface $\delta = I_\mathrm{r}^\mathrm{old}(\mathbf{l})_z$, we can reconstruct the world position of the reflected surface as

$$\mathbf{w}_\mathrm{r} = \mathbf{w}_\mathrm{d} + \delta\hat{r} \tag{1}$$

where $\hat{\mathbf{r}}$ gives vector $\mathbf{r}$ of unit length. These positions are combined in an image W_r.

Index of refraction For the refractive case, also the ratio of indices of refraction η is required. In many applications, with only one specular material, it is sufficient to assume η to be constant and known a priori, such as for "water" or "glass". In our implementation for the scene in Fig. 1 with more than one material, the IOR index is additionally transmitted.

Refracted position The refractive decoding for positions $\mathbf{f} \in \mathbb{R}^3$ is similar to the reflective one: $\mathbf{f} = \texttt{refract}(\mathbf{w}_\mathrm{d} - \mathbf{c}^\mathrm{old}, \mathbf{n}_\mathrm{d}, \eta)$, where `refract` is the OpenGL refraction function and $\eta \in \mathbb{R}$ is the ratio of IORs. $\mathbf{f}$ is then used in Eq. 1 instead of $\mathbf{r}$. These positions are combined in an image W_f.

Validity An additional single bit mask is used to identify the nodes of the ray tree that contributed to the pixel, e. g., diffuse and refractive node. This is important to later know, which areas need to be inpainted and which remain undefined. It is also used to reduce bandwidth by filling undefined RGBZ values with arbitrary positions, as well as it can be used to improve convergence speed of the specular flow in the vicinity of undefined areas. To save additional bandwith this information could also be packed into the RGBZ image using specific indicator values i. e., negative depth values.

Deeper ray tree decoding Deeper nodes of the ray tree are decoded as described above for the reflective and refractive case where the positions and normals are replaced by the respective parent nodes entries.

Transmitted data In summary the server transmits one RGBZ image per node of the ray tree, a single bit mask indicating the tree nodes contribution and the IOR index as well as the model view matrix used to create the ray tree.

For the RGBZ image we use 8 bits per color and 16 bit per depth channel which gives a total of 8 + 8 + 8 + 16 = 40 bits per pixel. The number of bits for the validity bit mask depends on the depth of the ray tree i. e., the maximum number of combined specular events in the tree, e. g., for a single reflection and refraction with a diffuse image we have a bit mask of two bits per pixel. The IOR bit mask depends on the number of different materials i. e., for two specular materials we have one bit per pixel. The modelview matrix consits of twelve 16 bit floating point numbers. In total we therefore would have to transmit $P \times (N \times 40 + D + M) + 12 \times 16$ uncompressed bits per frame, where P is the number of pixels, N is the number of nodes in the ray tree, D is the depth of the ray tree and M is the number of specular materials.

3.4. Warping

After decoding the last transmitted ray tree, the old image can be warped to resemble the view of the current model-view-projection matrix $\mathsf{M}^\mathrm{new} \in \mathbb{R}^{4\times4}$, i. e., a screen space flow from the old image can be computed, using only the last transmitted ray tree. In particular we do not need any information of the new frame except for the new modelview matrix which is created at the client-side anyway. The flow yields for every pixel in the new image an offset where to read in the old image. A distinction has to be made, whether the warp is to be diffuse, reflective or refractive as all three lead to different flow fields. A light-weight computation for those fields is the technical core of our technique. We again perform the calculations independently for each node and in pre-order starting at the root node. The flow calculated in each node is applied to the color, position and normal images of the node in the old image, to create the new image.

Diffuse flow The current screen space position $\mathbf{l}^\mathrm{new} \in \mathbb{R}^2$ of the position at location $\mathbf{l}$ in the old image can be found as $\mathbf{l}^\mathrm{new} = \texttt{project}(\mathsf{M}^\mathrm{new}, W_\mathrm{d})$, where `project` is the OpenGL projection function. The forward flow $f_\mathrm{d}^{-1} : \mathbb{R}^2 \to \mathbb{R}^2$ for each location of the old image to new the image is thus defined by $f_\mathrm{d}^{-1}(\mathbf{l}^\mathrm{old}) = \mathbf{l}^\mathrm{new} - \mathbf{l}^\mathrm{old}$.

However as we want to fill the new image by gathering, we are interested in the flow vectors at all locations of the new image, for which we need to find the backward flow f_d. The backward flow is not the negation of the forward flow and is only defined for some $\mathbf{l}$, namely the pixels that were visible in the old image.

We find this flow using a simplified version of the Iterative Image Warping algorithm [BMS*12]: All old image location values $\mathbf{l}^\mathrm{old}$ are splat to the new flow, as in $f_\mathrm{d} \leftarrow$

$\texttt{splat}(f_\mathrm{d}^{-1}(\mathbf{l}^\mathrm{old}), W_\mathrm{d}(\mathbf{l}^\mathrm{old})_z)$, where `splat` is the OpenGL point-primitive drawing with depth testing that takes as the first parameter the flow value to be splat and as the second parameter a z value used for depth testing. Regrettably, this approach leads to holes resulting from magnification (Fig. 4 a and b). As a solution, the flow at each pixel is replaced by the flow of the pixel that is closest to the camera in a 3×3 neighborhood. This closes holes, but leads to dilation of edges (Fig. 4 c). This again can be resolved in another step, performing fixed point iteration (FPI) described in [BMS*12] and discarding pixels where the convergence criterion is not satisfied (Fig. 4 d). The result is the diffuse backward flow $f_\mathrm{d}(\mathbf{l}^\mathrm{new})$ at all new images positions.

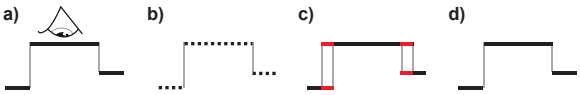

Figure 4: *Four steps of our warping. Each image shows the z-buffer with the camera at the top of the image.* a) *The old image.* b) *The new image after point splatting when the camera came closer. This image suffers from magnification holes.* c) *Hole-filling closes them but results in a dilation that enlarges nearby surfaces.* d) *One step of fixed-point iteration [BMS*12] detects and removes these issues.*

Specular flow Specular flow [AVBSZ07] is substantially more involved than diffuse flow. If we would assume only planar reflections with a constant normal, the backward flow could simply be computed by reflecting the world positions W_r on the reflection plane and inverting the flow as described for the diffuse flow. However for arbitrary reflective surfaces, e. g., curves, spheres, this does not hold any more. Even calculating the backward flow for a sphere involves solving a quadric and would also require to have a parametric description of the sphere. The refractive case is even more intricate. For all those reasons, we opt for a different, more general solution, handling reflection and refraction of arbitrary shapes. Next, we will describe the solution for general specular flow, and defer the specific details of reflection and refraction for later.

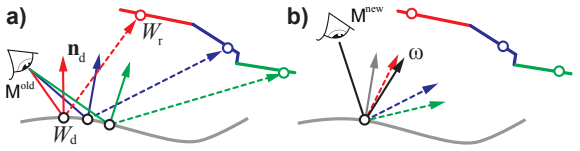

Figure 5: a): *Three pixels* (circles) *in the old view image with their diffuse positions W_d, their normals $\mathbf{n}_\mathrm{d}$ and their reflected positions W_r.* b): *A pixel in the new view* (circle). *The desired reflected direction* ω (black arrow) *is compared to the direction between the diffuse position and the reflected positions of the old image. The position with the most similar direction, (red), is selected as the best match* $\mathbf{p}^\star$.

For each location on the warped specular surface of the previous step we can calculate the specular ray direction ω by plugging in the warped diffuse position, normal and new camera position into the specular transport. Now, for an arbitrary world position $\mathbf{p}$ we can calculate how well the vector from $\mathbf{w}_\mathrm{d}$ to $\mathbf{p}$ agrees with the specular direction ω as $\Delta(\omega, \mathbf{p}) = \measuredangle(\omega, \mathbf{p} - \mathbf{w}_\mathrm{d})$ (Fig. 5). We want to find

$$\mathbf{p}^\star(\omega) = \operatorname*{arg\,min}_{\mathbf{p} \in W_\mathrm{s}} \Delta(\omega, \mathbf{p}),$$

i. e., the pixel $\mathbf{p}^\star$ in the specular position image W_s that "is in the right direction". The remaining question is how to perform this minimization?

We minimize the function using a slightly modified gradient descent procedure. Computing the analytical derivative of Δ w. r. t. $\mathbf{p}$ is not feasible as it would require an analytical surface representation unknown to the client. For this reason we use finite differences in image space to approximate the gradient of Δ. W_s has large undefined areas and sharp edges at depth discontinuities, for which calculating the image-space derivative is not feasible. If we detect such a case we simply reject the last step and choose a lower step size. If the distance Δ of the best found match to the specular direction ω is still above a certain threshold ε the pixel is set to be undefined. In all of our examples we use 10 gradient descent steps with a step size of 10^{-5} and a value of $\varepsilon = 1°$.

Gradient descent will perform much more efficiently given a good initial guess $\mathbf{p}_0$. As the initial guess, we will use a linear combination of two initial guesses: The first one is the diffuse flow f_d. The second is the specular initial guess f_s which is different for reflection and refraction to be defined later. The two initial guesses are blended based on their respective quality Δ, i. e., $\mathbf{p}_0 = \lambda f_\mathrm{d} + (1-\lambda) f_\mathrm{s}$, where $\lambda = 1 - \Delta_\mathrm{d}/(\Delta_\mathrm{d} + \Delta_\mathrm{s})$ with $\Delta_\mathrm{d} = \Delta(\omega, W_\mathrm{d}(\mathbf{l} + f_\mathrm{d}(\mathbf{l})))$ and $\Delta_\mathrm{s} = \Delta(\omega, W_\mathrm{s}(\mathbf{l} + f_\mathrm{s}(\mathbf{l})))$ (Fig. 6).

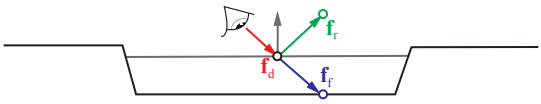

Figure 6: *Initial guesses for reflective and refractive warping: diffuse (red), reflective (green) refractive (blue).*

The initial reflection guess is a simple reflection around the current normal. The initial refractive guess is to treat the interface as transparent with an IOR ratio η of 1. More advanced initial guesses, especially for refraction, remain future work.

Our gradient descent algorithm assumes specular and diffuse surfaces that vary smoothly and do not have too many local optima that are close. However we found that for highly non-smooth surfaces the human observer has a hard time to correctly identify false solutions and that specular surfaces in the real world very often fulfill this criteria. For this reason

our results demonstrate rather smooth shapes such as planes and spheres where warping artifacts would become more objectionable than for complex geometry that would mask these artifacts. The solutions found by our gradient descent algorithm are not explicitly required to be temporally consistent and might differ from frame to frame. Additionally more than one optimal solution can potentially exist on complex shapes. However we found our coherent initial guess to steer the optimization to a coherent result in practice.

Deeper ray flows The flows for deeper nodes of the ray tree are performed accordingly as described before where the respective positions and normals are taken from the parent node instead of the root node. In our experiments we found that for deeper ray nodes the difference between correct and diffuse warped positions are not easily observable by humans and opt for using only the first reflection and refraction of the ray tree resulting in a ray tree with only three nodes: one diffuse, one reflective and one refractive node.

3.5. Final compositing

The disocclusions of the warping and the undefined pixels of the gradient descent optimization are filled using a hierarchical hole filling approach [GGSC96] on each node independently which skips pulling and pushing from and to invalid pixels. Finally, the different nodes are combined by addition, weighted by the Fresnel terms computed from the view position, the normals and the IORs.

4. Results

The effect of our specular warping in comparison to assuming a diffuse warp everywhere is shown in Fig. 7. The convergence of the optimization is seen in Fig. 8. The influence of the filter used to remove the artifacts of the normal reconstruction can be seen in Fig. 9. Other applications of specular novel-view image synthesis are shown in Fig. 10. Finally, the performance breakdown of the individual steps of our approach is shown in Tbl. 1.

As both diffuse [Koe86] and specular [AVBSZ07] optic flow are both temporal perceptual effects, we encourage the reader to see the supplemental video for a comparison of our approach to a naïve implementation. It can be seen that a naïve interpolation leads to disturbing inconsistencies. In our experience these result in an experience that objects visible along specular parts are deforming e. g., falling walls, bending grounds, etc.

5. Discussion and Conclusion

We proposed an approach to synthesize novel views of rendered images including specular reflections and refractions. This was achieved by splitting the scene rendering into an RGBZ ray tree, where each node is an image that is warped

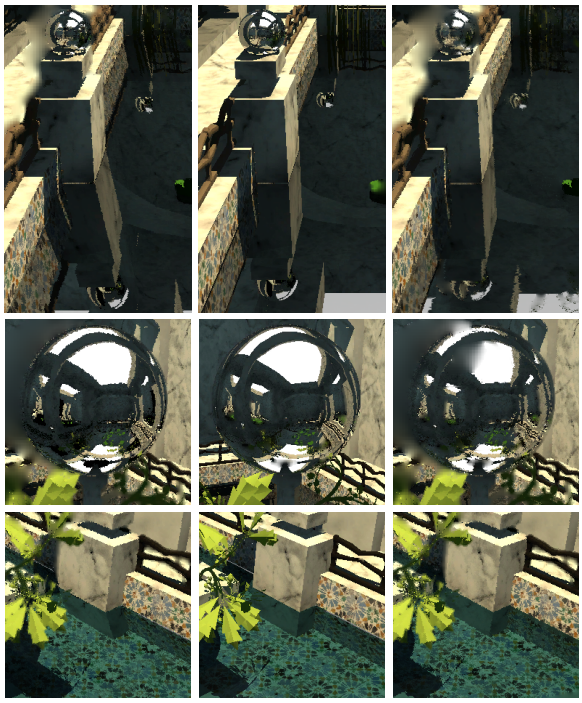

Figure 7: *Result when performing a left movement.* Top: *Reflections on a planar surface.* Center: *Reflections on a sphere.* Bottom: *Refractions on a planar surface.* From left ro right*: Naïve diffuse warping, ground truth, our results.*

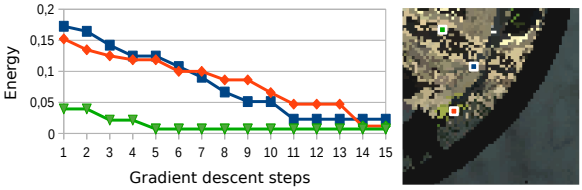

Figure 8: *Energy at the marked pixels over the steps of the gradient descent for reflections on a sphere. Pixels in the center of the sphere profit from a more accurate initial guess.*

independently using a fast approximative reflective and refractive flow. The results synthesize novel output images with very low latency from input image streams with substantial latency. Applications include remote rendering and head-mounted displays, where latency is to be avoided.

Our method is subject to certain limitations. First, our approach creates derivatives from compressed images, which should be avoided (Fig. 11). A more adapted compression and blurring scheme would be required to strike the right balance between quality and performance. For high-quality results at medium bandwidth, the normals could be encoded [MSS*10] and transmitted as well. Splitting the image into multiple, pre-multiplied images has the positive consequence, that faint

Figure 10: *The naïve approach* (top) *compared to ours* (center) *for computing depth-of-field a stereo-image pair from a single ray tree. A slight translation and rotation is added to emphasize the difference to our results.* Bottom: *Both effects combined.*

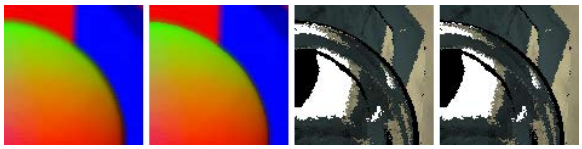

Figure 9: *Influence of normal filter on the warping results.* From left to right: *Normals filtered with gaussian filter, normals filtered with bilateral filter, warping results for gaussian normals, warping results for bilateral normals*

Table 1: *Computation time on a GeForce GTX 765M. Reflection and refraction decoding are combined into "Decoding Specular" as they can be performed in a single shader.*

Stage	Node		Time
Decoding	Diffuse		0.48 ms
	Normal	+	1.01 ms
	Specular	+	0.96 ms
Warp	Diffuse	+	2.49 ms
	Reflection	+	4.09 ms
	Refraction	+	3.70 ms
Hole filling	Diffuse	+	1.75 ms
	Reflection	+	1.47 ms
	Refraction	+	1.24 ms
Compositing		+	1.14 ms
Total			18.37 ms

signals results in low values and compress more as well as the fact that signals get de-correlated more. Adaptive coding, including masking between nodes, as well as temporal masking between nodes seems worth exploration: a reflection on a high-contrast texture can be compressed more than a floor with low contrast. Assigning a single depth value to each pixel does not go well together with antialiasing. Instead, an approach similar to FXAA needs to be run as a post-process. Finally, distribution effects and participating media do not map well to a ray tree as they decouple radiance and depth in a general way. Handling such effects will require an entirely new approach, a promising avenue of further research.

References

[AH95] ADELSON S. J., HODGES L. F.: Generating exact ray-traced animation frames by reprojection. *Comp. Graph. & App., IEEE 15*, 3 (1995), 43–52. 2

[And10] ANDREEV D.: Real-time frame rate up-conversion for video games: Or how to get from 30 to 60 fps for "free". In *ACM SIGGRAPH Talks* (2010), pp. 16:1–16:1. 2

[AVBSZ07] ADATO Y., VASILYEV Y., BEN-SHAHAR O., ZICKLER T.: Toward a theory of shape from specular flow. In *Proc. ICCV* (2007), pp. 1–8. 5, 6

[BJ88] BADT JR S.: Two algorithms for taking advantage of

Figure 11: Top: *Impact on the warping quality caused by JPEG-compressed depth.* Bottom: *Uncompressed depth.*

temporal coherence in ray tracing. *The Visual Computer 4*, 3 (1988), 123–132. 2

[BMS*12] BOWLES H., MITCHELL K., SUMNER R. W., MOORE J., GROSS M.: Iterative image warping. In *Comp. Graph. Forum (Proc. Eurographics)* (2012), vol. 31, pp. 237–46. 2, 4, 5

[Che97] CHEVRIER C.: A view interpolation technique taking into account diffuse and specular inter-reflections. *The Visual Computer 13*, 7 (1997), 330–41. 2

[COMF99] COHEN-OR D., MANN Y., FLEISHMAN S.: Deep compression for streaming texture intensive animations. In *Proc. SIGGRAPH* (1999), pp. 261–7. 2

[CW93] CHEN S. E., WILLIAMS L.: View interpolation for image synthesis. In *Proc. ACM SIGGRAPH* (1993), pp. 279–88. 2

[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. In *Computer Graphics Forum (Proc. Eurographics)* (2010), vol. 29, pp. 713–22. 2

[DRE*10] DIDYK P., RITSCHEL T., EISEMANN E., MYSZKOWSKI K., SEIDEL H.-P.: Adaptive image-space stereo view synthesis. In *Proc. VMV* (2010), pp. 299–306. 2

[DWS*88] DEERING M., WINNER S., SCHEDIWY B., DUFFY C., HUNT N.: The triangle processor and normal vector shader: a VLSI system for high performance graphics. In *Proc. SIGGRAPH* (1988), vol. 22, pp. 21–30. 3

[GGSC96] GORTLER S. J., GRZESZCZUK R., SZELISKI R., COHEN M. F.: The Lumigraph. In *Proc. SIGGRAPH* (1996), pp. 43–54. 6

[Koe86] KOENDERINK J. J.: Optic flow. *Vis Res 26*, 1 (1986), 161–79. 6

[Lev95] LEVOY M.: Polygon-assisted JPEG and MPEG compression of synthetic images. In *Proc. SIGGRAPH* (1995), pp. 21–8. 2

[LR98] LISCHINSKI D., RAPPOPORT A.: Image-based rendering for non-diffuse synthetic scenes. In *Rendering Techniques*. 1998, pp. 301–14. 2

[MMB97] MARK W. R., MCMILLAN L., BISHOP G.: Post-rendering 3D warping. In *Proc. ACM I3D* (1997), pp. 7–ff. 2

[MSS*10] MEYER Q., SÜSSMUTH J., SUSSNER G., STAMMINGER M., GREINER G.: On floating-point normal vectors. In *Comp. Graph. Forum* (2010), vol. 29, pp. 1405–9. 6

[PEL*00] POPESCU V., EYLES J., LASTRA A., STEINHURST J., ENGLAND N., NYLAND L.: The WarpEngine: An architecture for the post-polygonal age. In *Proc. ACM SIGGRAPH* (2000), pp. 433–42. 2

[PHE*11] PAJAK D., HERZOG R., EISEMANN E., MYSZKOWSKI K., SEIDEL H.-P.: Scalable remote rendering with depth and motion-flow augmented streaming. In *Computer Graphics Forum (Proc. Eurographics)* (2011), vol. 30, pp. 415–24. 2, 3

[PHM*14] PAJĄK D., HERZOG R., MANTIUK R., DIDYK P., EISEMANN E., MYSZKOWSKI K., PULLI K.: Perceptual depth compression for stereo applications. *Computer Graphics Forum (Proc. Eurographics) 33*, 2 (2014). 2, 3

[SGHS98] SHADE J., GORTLER S., HE L.-W., SZELISKI R.: Layered depth images. In *Proc. ACM SIGGRAPH* (1998), pp. 231–42. 2

[SKG*12] SINHA S., KOPF J., GOESELE M., SCHARSTEIN D., SZELISKI R.: Image-based rendering for scenes with reflections. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4 (2012). 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 *EG STAR* (2011), pp. 101–126. 2

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Commun. ACM 23*, 6 (1980), 343–349. 2

[Wol98] WOLBERG G.: Image morphing: a survey. *The visual computer 14*, 8 (1998), 360–72. 2

[YTS*11] YANG L., TSE Y.-C., SANDER P. V., LAWRENCE J., NEHAB D., HOPPE H., WILKINS C. L.: Image-based bidirectional scene reprojection. In *ACM Trans. Graph.* (2011), vol. 30, p. 150. 2