Real-time Novel-view Synthesis for Volume Rendering Using a Piecewise-analytic Representation

G. Lochmann\textsuperscript{1} B. Reinert\textsuperscript{2} A. Buchacher\textsuperscript{1} T. Ritschel\textsuperscript{3}

\textsuperscript{1}Universität Koblenz-Landau \textsuperscript{2}MPI Informatik \textsuperscript{3}University College London

Figure 1: Starting from an original volume rendering (left) our approach generates novel views in milliseconds (right).

Abstract

Novel-view synthesis can be used to hide latency in a real-time remote rendering setup, to increase frame rate or to produce advanced visual effects such as depth-of-field or motion blur in volumes or stereo and light field imagery. Regrettably, existing real-time solutions are limited to opaque surfaces. Prior art has circumvented the challenge by making volumes opaque i.e., projecting the volume onto representative surfaces for reprojection, omitting correct volumetric effects. This paper proposes a layered image representation which is re-composed for the novel view with a special reconstruction filter. We propose a view-dependent approximation to the volume allowing to produce a typical novel view of $1024 \times 1024$ pixels in ca. 25 ms on a current GPU. At the heart of our approach is the idea to compress the complex view-dependent emission-absorption function along original view rays into a layered piecewise-analytic emission-absorption representation that can be efficiently ray-cast from a novel view. It does not assume opaque surfaces or approximate color and opacity, can be re-evaluated very efficiently, results in an image identical to the reference from the original view, has correct volumetric shading for novel views and works on a low and fixed number of layers per pixel that fits modern GPU architectures.

1. Introduction

Illuminating surfaces can be costly for complex global-illumination shaders, detailed geometry and natural lighting. Fortunately, images are often coherent across time and space: they change little between different viewpoints, and therefore, many interactive applications make use of novel-view synthesis to re-project a shaded image to a new view instead of rendering it. This allows to quickly compute latency-compensated image streams, stereo pairs or light fields and can substantially reduce the cost of depth-of-field or motion blur. Producing novel views can be done in a few milliseconds on modern graphics hardware.

Even more costly than surface shading is volume lighting. Consequently, the demand for novel-view synthesis is even more pressing. Regrettably, no such procedure to generate novel views of volume-rendered scenes exists.

In this work, we propose a technique to do so. We will extend volume rendering based on emission-absorption ray-marching to produce a custom layered image representation. This representation allows to quickly produce views different from the original one. The representation is based on a piecewise approximation of the emission-absorption from the original view. It allows for a particularly simple and fast re-projection solution, yet, with proper inter-layer and inter-view sampling. Going beyond simple view-dependent encoding of the original emission-absorption information, our approach assures that the solution of the volume rendering equation using the compact representation produces exactly the same
image which the original representation would produce when seen from the original view. Typical volumes with many complex layers and transparent effects can be encoded into four to eight slices.

2. Related work

Volume rendering Volume rendering synthesizes images of scenes where light is not only reflected at discrete surfaces but also by particles distributed in a 3D medium [DCH88]. Such images have applications ranging from computer games over feature film production to scientific and medical visualization. Different solutions such as front-to-back and back-to-front compositing can solve the volume rendering equation (VRE) [Max95]. Using modern graphics hardware, direct volume rendering based on ray-marching is now widely available [KW03].

To accelerate volume rendering, different suggestions were made. One option is to re-use information from ray-casting the previous image to accelerate generation of new images [KSSE05]. Another popular one is to decompose the volume in a pre-process into planar layers called “Slabs” [CCF94]. Later, the layers can be quickly composed. Our approach is similar but produces the slices on-the-fly by ray-marching and storing an intermediate, non-planar volume representation that assures that both the original view can be reconstructed without error and novel views with a low error.

Another improvement often used is pre-integration [EKE01]: Instead of evaluating the integral of the transfer functions applied on the linear interpolation of two values, the result of this integration for all pairs of two values before applying the transfer function is cached. We also pre-integrate when generating the original view – be it in the presence of transfer functions or not – and use this information to improve speed and quality in a later step. Different from previous work, first, our calculation is done for every pixel at every frame, and second, the quantity integrated is converted back into emission and absorption that can be analytically integrated to produce novel views using a closed-form solution.

Recently, complex shading [HLSR08] such as indirect lighting in volumes has become a topic of interest. Full shading, e.g., using Monte Carlo [PHO4, RS07] or Photon Mapping [JC98] is expensive and has complex mathematical formulations [Max95]. Our approach is basically agnostic to the global illumination solution used as long as it can be computed once for a single view point.

Re-using shading computations for new view points has been used to accelerate rendering. Deep shadow maps allow to cast shadows in scenes with transparent structures [LV00, KN01, HKS06], pre-computed radiance transfers [SKS02] can be used for volumetric effects [Rit07], radial basis functions can capture smooth indirect volume lighting [ZRL08] and radiance caching was extended to volumes [JDZJ08]. All these approaches address how to re-use the shading, visibility or light transport term relating to $\kappa$ in Sec. 3.1 in 3D, but do not solve the VRE towards the 2D pixel itself. Properly caching this result is critical to combining detailed volumes that might not even fit device memory, for low latency in novel-view synthesis and for image quality.

Encoding VRE results from the camera is similar to what deep shadow maps [LV00] do from the light’s point of view [KN01, HKS06]. However, we do not suggest a way to re-evaluate emission or absorption from the light, but from a novel view.

Novel-views Producing novel views from novel view points for 3D-rendered images was proposed by Mark et al. [MMB97]. Typically, first a buffer with one unique depth value per pixel is rendered. Two main variants exist: scattering-type [MMB97] and gathering-type computation [BMS’12]. Our approach falls into the gathering category, however, with the generalization to not only finding a unique pixel in the input that maps onto the output pixel, but to enumerate an entire distribution of colors along a view ray. Things get more complex in the presence of transparency: Typically, here, images are decomposed into layers [SGHS98], which is obvious for opaque diffuse surfaces, but more involved in the general case. For transparent reflecting or refracting surfaces solutions have been proposed [LKR’14, ZR15], but these are limited to discrete surfaces at discrete depths while our paper is addressing the fully volumetric case. The fully volumetric case is the most general one. It is much sought for in remote rendering and Tomland et al. [THRS’01] have named the latency we address as a key practical challenge. When bandwidth is of no concern and the volume is static, a light field can be pre-computed from multiple view points and quickly ray-cast with depth correction to show volume data sets of opaque surfaces or multiple layers from a restricted set of views at high speed and quality [RSTK08]. Two approaches have considered producing novel views of volumes: the one of Mueller et al. [MSHC99] and the one of Zellmann et al. [ZAL12]. We compare our results to these solutions and find it to outperform them in all cases. Only if the volume rendering converges to a opaque surface, which is well-solved by classic methods [MMB97], all methods converge to a similar quality and speed.

Mueller et al. [MSHC99] use the classic slab-based volume rendering for novel-view synthesis. They suggest placing equidistant slabs [CCF94] with a coarse $(16 \times 16)$ field height that is later warped as a surface. Their results demonstrate applications to volumetric data but only for transfer functions that result in rather opaque surfaces. Even under this condition, substantial gaps between slabs appear (Fig. 5 in [MSHC99]), and even for solid surfaces. Our approach does not show such gaps and, using less layers, produces better images across the entire spectrum from opaque to transparent surfaces. A similar approach has been used to render on multiple devices and compensate for the latency between them [HKS10].

In the approach of Zellmann et al. [ZAL12] the volume rendering is assumed to be reducible to a single opaque surface for the purpose of re-projection. Here, every pixel has a depth and is drawn as a point, instead of using a course grid of quads. Zellmann et al. propose different heuristics to find the depth of these representative surfaces, such as mean, min, max of depth etc. and conclude mean depth to be a good solution for their examples. The assumption of a single depth will work for volume data sets and transfer functions that have a single dominant depth, while our solution spans the entire range between opaque and transparent media.

3. Layered Volume Re-projection

Overview Our approach has two main components (Fig. 2): The first one synthesizes a view-dependent volume representation of the
original view’s image (Fig. 2, c), the second one re-synthesizes an image for a novel view from that representation (Fig. 2, d). Both steps fit modern GPUs well. In a typical remote rendering setup, the first step occurs on the server, the second on the client.

The old-view synthesis performs one pass of volume rendering from the original view, including full shading and accessing the detailed 3D volume. It produces \( k \) depth layers for every pixel. Every pixel stores an RGBA color and a depth.

The re-synthesis uses this representation to produce an image from a novel view. It performs a heavily simplified ray-marching that makes use of the information available from the original view to produce a new one. In particular, we suggest to 2D-ray march the epipolar back-projection of the novel-view ray in the original-view.

Layering makes use of the ideas from the emission-absorption approximation we propose, so it will be described first (Sec. 3.1) and layering will be described later (Sec. 3.2). At run-time the steps are executed in the opposite order: first layers are found, then the volume inside is approximated. In any case, novel-view synthesis happens and is described last (Sec. 3.3).

### 3.1. Original view synthesis

The original view synthesis produces an RGBA image and a depth value for each pixel. This representation is constructed for all pixels independently and in parallel on a GPU.

For every original-view image pixel from view \( V_1 \) exists a ray \( r_1 \). This ray is divided into \( n \) segments in a way to be defined later. Let \( s_1 \) and \( s_2 \) be the start and end distance of one such segment, respectively.

The volume rendering equation states that the radiance at a position \( \mathbf{x} \) from a direction \( \mathbf{w}_i \) coming from the section \( s_1, s_2 \) is

\[
L(\mathbf{x}, \mathbf{w}_i, s_1, s_2) = \int_{s_1}^{s_2} \eta(s) \kappa(s) ds,
\]

where \( \eta(s) \) and \( \kappa(s) \) are the integral of the product of emission \( \eta \) and transmittance \( \kappa \) between \( s_1 \) and \( s_2 \). Transmittance between the sensor \( \mathbf{x} \) and the point \( y(s) \) along the ray is

\[
\kappa(s) = \exp(-\int_0^s \tau(y(t)) dt)
\]

where \( \tau \) is the extinction that is high if the medium is dense. Emission combines both reflected and initially emitted light and is denoted as

\[
\eta(s) = L_c(y(s), -\omega_o) + \int_\Omega p(y(s), -\omega_o, \omega_i) L_c(y(s), \omega_o, 0, \infty) d\omega_o,
\]

where \( L_c \) is initial emission, and \( p \) is the scattering function. The radiance \( L \) is associated with the RGB color \( C \) of an image pixel; the pixel opacity \( T \) (alpha value) with \( \kappa(\infty) \).

The view encoding is assumed to be able to evaluate \( \eta \) using
whatever approach available to compute (global) light transport in a medium. In a client-server setting, a fast server is assumed to spend substantial computational power requiring energy and cooling on the evaluation that would not be available to the client.

The client will avoid solving the integral to save computational resources. Instead, it will make use of a simplified equation for re-projection that assumes emission \( E \) and absorption coefficient \( A \) to be constant along the ray segment. Denoting the length of the ray segment as \( z = z_2 - z_1 \), this results in the Beer-Lambert equations for homogeneous media

\[
C = ET \quad \text{and} \quad T = \exp(-Az).
\]

It is tempting to use the average emission or opacity values

\[
\bar{E} = \frac{1}{z} \int_{s_1}^{s_2} \eta(y(s)) ds \quad \text{resp.} \quad \bar{A} = \frac{1}{z} \int_{s_1}^{s_2} \tau(y(s)) ds.
\]

However, solving the VRE with average emission or opacity values \( \bar{E} \) and \( \bar{A} \) will not result in \( C \) and \( T \), not even from the original view, due to the non-linearity of the VRE.

Instead, we solve for a per-ray segment emission and absorption value \( E \) and \( A \), such that when volume-rendering a ray segment of length \( z \) from the original view, the original color \( C \) and opacity \( T \) are produced. This is achieved by re-arranging for \( E \) and \( A \) when \( C \), \( T \) and \( z \) are given:

\[
E = \frac{C}{T} \quad \text{and} \quad A = -\log(T)/z.
\]

Using these values, analytical re-rendering from the original view produces the original values, but also allows to properly render from other views, where \( z \) can be very different from the original view. This ability is key both to our quality and speed: Without a proper formulation, intervals need to be short as they do not allow extrapolation to very different views. Many short intervals are expensive to compute. Typically the intervals are much larger in depth than a pixel is wide. When seen “from the side” the values need to be corrected for, including the non-linearities.

### 3.2. Layering

Layering selects \( k \) depth values \( d_i \) for each ray, one for every ray segment \( i \). This is done independently between pixels by inverting transmittance in regular steps. Formally, let \( F(d) = \kappa(d)/\kappa(\infty) \) be the normalized transmittance, where the term \( \kappa(\infty) \) is a normalization for pixels that do not reach full transmittance at any finite depth. We now look for values \( d_i \) that have accumulated a fraction \( y_i = (i+0.5)k^{-1} \), such that \( F(d_i) = y_i \). This can be achieved in two passes along the ray: The first computes the normalization \( \kappa(\infty) \). The second marches the ray, updating the CDF value \( F(d) \) at every location \( d \) and producing \( d_i \) at the first occasion where \( F(d_i) \geq y_i \).

### 3.3. Novel view synthesis

Novel views are produced by efficient ray-marching on the view-dependent approximate representation devised above. Instead of making many steps on many voxels and computing emission using an expensive shading, we will now make very few steps, on very few voxels and on cached shading results. From view \( \mathcal{V}_2 \), again rays are cast and the color is reconstructed independently and in parallel. We will now discuss reconstructing the color for a ray \( r_2 \) through an arbitrary pixel.

First, \( r_2 \) is projected into the old view (Fig. 4, a). While there is a single pixel for every ray in the novel view, the same ray spans many pixels in the old view. Let \( p_{\text{start}} \) and \( p_{\text{end}} \) be the 2D start and end pixel locations of the epipolar line from \( r_2 \) in the old view \( \mathcal{V}_1 \). Starting at \( p_{\text{start}} \) we enumerate all pixels in the original image that intersect the 2D line.

Each pixel forms a perspective frustum in the original view (Fig. 4, b). At one pixel, we find the layer pair \( (d_i, d_{i+1}) \) in which the ray enters, and continue with \( (d_{i+1}, d_{i+2}) \), until the ray leaves the frustum again. Now, for every segment \( (d_i, d_{i+1}) \), and the novel-view ray, there is an entry and an exit position \( x_i \) and \( x_{i+1} \). The contribution on the length \( z_i = \|x_i - x_{i+1}\| \) can be found analytically as explained above (Eq. 1): As we know emission \( E \) and absorption \( A \) that is constant between \( x_i \) and \( x_{i+1} \) we can compute the color \( C \) and transmittance \( T \) along the new ray in closed form. The segments are composed front-to-back (Fig. 4, c) in an inner loop and front to back along the 2D marching (not shown in Fig. 4).

The loop over the layers has either to be forward, when seeing it from the front, or backward, when seeing it from behind. The cases are separated by computing a dot product of old and new view direction.

### 4. Results

Here, we report qualitative (Sec. 4.1) and quantitative (Sec. 4.2) evaluation of our result, including comparisons to prior work and re-rendered references. Finally, we demonstrate several novel applications enabled by our approach (Sec. 4.3).

We refrain from reporting any speed-up over re-shading, which can be made arbitrarily favorable for any re-using approach by simply choosing sufficiently expensive shading. Our examples uses Monte Carlo volume shading from natural illumination i.e., environment maps, which requires many seconds for each image, indicting speed-up of several orders of magnitude.
As the client constantly changes its pose, it permanently requires im-
we use six scenes: a synthetic cube scene, four typical CT scans and
Typical results of our approach are seen in Fig.
4.1. Qualitative results
Typical results of our approach are seen in Fig. 10. We see that our
approach handles both homogeneous and heterogeneous volumes,
producing the right color and transparency from novel views. It
allows for different types of volumes, including abstract, medical
or game-like 3D volumes. Both non-dense, transparent and dense,
opaque, surface-like media are possible.
Our approach produces novel views in a few milliseconds. Both
speed and quality depend mainly on the difference between the old
and the new view as seen in Table 2. Quality is reduced, as large
changes are harder to compute, which is similar for opaque surfaces.
4.2. Quantitative results
Here we analyze the ability of our approach to predict novel views
by comparing the images produced by our approach to images
produced using a full re-rendering of the novel view. As test data,
we use six scenes: a synthetic cube scene, four typical CT scans and
a cloud as found in computer games. Images are compared using
DSSIM [WBSS04] where a smaller value is better. Comparison is
done for extrapolation across three different degrees of “novelty” of
the novel view, measured in degree between the view directions (5°,
10° and 15°). The quality of novel views is shown in Table 2.
Additionally, we compare it to the approach of Zellmann et al.
[ZAL12] and the approach of Mueller et al. [MSHC99]. While their
approaches can be faster, the quality is consistently worse than ours.
Additionally, we show a visual comparison of our approach with
different numbers of layers in Fig. 6. We conclude, that our approach
produces novel views with much higher quality at similar speed.
4.3. Applications
Novel-view synthesis has many applications. We here demonstrate
for the first time novel-view synthesis-based remote, stereo and
distribution-effect renderings of volumes.
Remote rendering Currently, the premiere application of novel-
view synthesis is latency compensation and stereo synthesis for
mobile rendering, in particular, for head-mounted displays where
latency can have adverse effects on viewing comfort. In such a sce-
nario, images are rendered on a server and sent to a mobile client.
As the client constantly changes its pose, it permanently requires im-
ages that are similar, but slightly different from the transmitted pose.
Our approach can compensate for this difference – and therefore
the latency – by producing the view required. It can also be used
to produce a stereo image pair from a monocular stream. A typical
example would be a game scenario of a flight simulator involving
clouds as seen in Fig. 5.
For remote rendering, the images transmitted need to be com-
pressed in practice. The effect on the final image when applying
JPEG 2000 image compression to each individual layer at a resolu-
tion of 1024×1024 and k = 2 with 50 frames per second is seen in
Fig. 7. For every layer, we need to transmit an RGB E and in mono
an A and a depth value z. Instead of transmitting E = C/T , that can
have a high dynamic range, C is transmitted and E is re-computed
from T and Z on the client.
Stereo rendering A typical use of novel views is to turn one mono-
scopic image into a stereo image pair (Fig. 8, right). Previously,
this was limited to opaque surfaces or required re-shading and re-
arching the volume.
Distribution effects Distribution effects can use novel-view syn-
thesis to convert a single image into many other images that provide
samples across the lens or time domain. Averaging these results in
depth-of-field (or motion blur), that is used to apply context in visu-
alization [SPG*M11], as seen in Fig. 8, left. Previously, such images
could not be produced without costly ray-marching and shading.
5. Discussion
Here we would like to discuss some important differences of our
algorithm compared to previous work and potential alternatives.
Previous work In terms of previous work, our approach is able
to produce novel views of volumes, while the approaches of
Mueller et al. [MSHC99] and Zellmann et al. [ZAL12] were only
demonstrated on volumetric data sets that effectively result in opaque
surfaces. This relation is seen in Fig. 9.
Alternative solutions We have explored alternative algorithms that
also use forward splatting and simpler formulations based on com-
posing. One such approach could just store the effective per-pixel
color and transparency of every layer and compose these front-to-
back from novel views. We did not consider such an approach for
three reasons: First, color or transparency cannot predict well, how
a ray segment would appear from a different view: If the original
Table 1: Compute time (in ms, lower is better) and DSSIM error (lower is better) of our approach for different degrees of view novelty, numbers of layers (columns) and scenes (rows) seen in Fig. 1 (Zaedyus) and Fig. 10 (other). 1024 × 1024 on a Nvidia GeForce GTX 980.

<table>
<thead>
<tr>
<th></th>
<th>2 Layer Error Speed</th>
<th>4 Layer Error Speed</th>
<th>8 Layer Error Speed</th>
<th>2 Layer Error Speed</th>
<th>4 Layer Error Speed</th>
<th>8 Layer Error Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cubus</td>
<td>.003 21.8</td>
<td>.002 23.4</td>
<td>.002 27.5</td>
<td>.007 23.4</td>
<td>.005 25.8</td>
<td>.003 32.9</td>
</tr>
<tr>
<td>Zaedyus</td>
<td>.035 22.7</td>
<td>.024 24.3</td>
<td>.016 29.0</td>
<td>.049 24.5</td>
<td>.031 27.6</td>
<td>.020 35.6</td>
</tr>
<tr>
<td>Ictiobus</td>
<td>.033 23.7</td>
<td>.018 25.6</td>
<td>.011 29.4</td>
<td>.055 24.6</td>
<td>.027 28.6</td>
<td>.015 37.4</td>
</tr>
<tr>
<td>Celestus</td>
<td>.052 23.4</td>
<td>.024 26.7</td>
<td>.017 34.4</td>
<td>.072 26.3</td>
<td>.036 31.9</td>
<td>.023 45.3</td>
</tr>
<tr>
<td>Cloud</td>
<td>.017 23.2</td>
<td>.010 26.7</td>
<td>.017 32.9</td>
<td>.022 25.9</td>
<td>.013 30.8</td>
<td>.007 40.8</td>
</tr>
<tr>
<td>Walnut</td>
<td>.060 22.2</td>
<td>.041 24.1</td>
<td>.033 28.7</td>
<td>.083 24.2</td>
<td>.052 27.7</td>
<td>.039 35.5</td>
</tr>
</tbody>
</table>

Table 2: Time for producing a ten-degree novel view (in ms, lower is better) and DSSIM error (lower is better) of our (k = 4) and other approaches (columns) for different scenes (rows).

<table>
<thead>
<tr>
<th></th>
<th>Ours Error Speed</th>
<th>[ZAL12] Error Speed</th>
<th>[MSHC99] Error Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cubus</td>
<td>.005 25.8 ms</td>
<td>.010 3.0 ms</td>
<td>.006 36.8 ms</td>
</tr>
<tr>
<td>Zaedyus</td>
<td>.034 27.6 ms</td>
<td>.081 3.0 ms</td>
<td>.041 33.5 ms</td>
</tr>
<tr>
<td>Ictiobus</td>
<td>.027 28.6 ms</td>
<td>.070 3.0 ms</td>
<td>.022 33.6 ms</td>
</tr>
<tr>
<td>Celestus</td>
<td>.036 31.9 ms</td>
<td>.067 3.0 ms</td>
<td>.070 40.0 ms</td>
</tr>
<tr>
<td>Cloud</td>
<td>.013 30.8 ms</td>
<td>.064 3.0 ms</td>
<td>.036 34.4 ms</td>
</tr>
<tr>
<td>Walnut</td>
<td>.052 27.7 ms</td>
<td>.109 2.9 ms</td>
<td>.063 34.4 ms</td>
</tr>
</tbody>
</table>

Figure 6: A reference, results of our approach at different numbers of layers k = {8, 4, 2} and the source image.

Figure 7: The effect of image compression (x axis, log-plot) on the image quality (y axis) (DSSIM, lower is better).

Figure 8: Stereo image synthesis (left) and volumetric depth-of-field rendering (right) using our approach (k = 4).

assumming the emission and absorption was constant. This allows for a fixed and small number of segments at each pixel. Second, composing from a novel view would require sorting the ray segments. Sorting is difficult on GPUs and typically done on the CPU, requiring CPU-GPU synchronization. Finally, our representation can be evaluated without drawing any new geometry in any specific order (splatting, or scatter-type computation). Instead, a new view is produced using the preferred gather-type computation by sampling epipolar projections of the novel ray segment in the representation.

Our layering scheme is only one out of several different alternatives. It has a few useful properties: it produces the same number of outputs, which is easily produced in parallel and later efficiently traversed. It puts details along the ray where it actually matters in terms of contribution to the original image, not just where the
We have presented a new approach to producing images showing novel views of volumes. This is often desired, but introduces the classic surface rendering problem of occlusions into volume rendering. When a medium is seen from a new view, the emission does not change. For many relevant objects, such as the silver lining on clouds this can be a visually important effect. Our approach could be extended to support such effects by representing the direction-dependent emission using Henyey-Greenstein functions such as previously done for rendering of volumetric lighting [ERWS12].

A totally different approach would be to learn how to produce novel views from example data [RFF14]. This could be one step towards producing novel views from clouds without modifying the volume rendering code, i.e., directly from photographs.

Acknowledgements We would like to thank the Digital Morphology at the University of Texas for providing some of the data sets.

6. Conclusion

We have presented a new approach to producing images showing novel views of volumes. To this end, we adapt the volume rendering to produce multiple layers of emission and absorption, such that their re-ray-casting produces the same result, yet, it can be efficiently ray-cast also from other views to produce high-quality predictions without re-shading or access to the full 3D data set. Typical applications are remote-rendering systems, either for entertainment in computer games, but equally well for serious applications such as remote-rendering of a detailed and high-quality shaded volume data set for a patient [THRS01] to be inspected in-situ using a mobile device with limited bandwidth and compute power.

Our main limitation is to assume an isotropic phase function: When a medium is seen from a new view, the emission does not change. For many relevant objects, such as the silver lining on clouds this can be a visually important effect. Our approach could be extended to support such effects by representing the direction-dependent emission using Henyey-Greenstein functions such as previously done for rendering of volumetric lighting [ERWS12].

Figure 9: Different approaches (rows) on different densities (columns), ranging from very transparent to surface-like. Density decreases from left to right. The DSSIM (lower is better) is shown in orange. The first row is the input image; the second and third competing approaches [ZAL12] and [MSH99] the fourth our approach (k = 4) and the fifth one as a reference.
Figure 10: Results of our approach ($k = 4$) for different views (row two, three and four) from an original volume rendering (first row)


