Fast Differentiable Transient Rendering for Non-Line-of-Sight Reconstruction (Supplementary Material)

Markus Plack Clara Callenberg Monika Schneider Matthias B. Hullin
University of Bonn
Bonn, Germany
{mplack,callenbe,hullin}@cs.uni-bonn.de, moschn@uni-bonn.de

In **Section 1** we explain the forward model and the derivation of the backward pass for both the confocal scanning setup and the exhaustive case. **Section 2** gives more details on the reconstruction algorithms and **Section 3** lists the experiments with their respective timings. Finally, **Section 4** shows intermediate steps of the depth map and albedo optimization for one reconstruction of the statue, and a single optimization of a transform with a larger amount of translation and rotation.

1. Derivations

We introduce the notation and common functions used here in Section 1.1. Section 1.2 gives a detailed derivation of the gradients from our forward model for the confocal case and Section 1.3 outlines the necessary changes when applying our method to simulate a non-confocal measurement.

1.1. Foundations

Here, we define functions used later on and give their derivatives. More symbols and their definition are given in Table 1.

Normal Vector The (unnormalized) normal vector of a triangle.

$$n(t_i) = (v_{i,1} - v_{i,0}) \times (v_{i,2} - v_{i,0}) \tag{1}$$

$$J_n(t_i) = \begin{pmatrix} 0 & (v_{i,2}-v_{i,1})_z & -(v_{i,2}-v_{i,1})_y \\ -(v_{i,2}-v_{i,1})_z & 0 & (v_{i,2}-v_{i,1})_x \\ (v_{i,2}-v_{i,1})_y & -(v_{i,2}-v_{i,1})_x & 0 \\ 0 & -(v_{i,2}-v_{i,0})_z & (v_{i,2}-v_{i,0})_y \\ (v_{i,2}-v_{i,0})_z & 0 & -(v_{i,2}-v_{i,0})_x \\ -(v_{i,2}-v_{i,0})_y & (v_{i,2}-v_{i,0})_x & 0 \\ 0 & (v_{i,1}-v_{i,0})_z & -(v_{i,1}-v_{i,0})_y \\ -(v_{i,1}-v_{i,0})_z & 0 & (v_{i,1}-v_{i,0})_x \\ (v_{i,1}-v_{i,0})_y & -(v_{i,1}-v_{i,0})_x & 0 \end{pmatrix} \tag{2}$$

Centroid The centroid of a triangle.

$$c(t_i) = \frac{1}{3}(v_{i,0} + v_{i,1} + v_{i,2}) \tag{3}$$

$$\nabla_{t_i} c(t_i) = \frac{1}{3}\mathbf{1}_9 \tag{4}$$

Symbol	Definition
$I(\mathcal{T})$	Rendering function of the transient image $I(\mathcal{T}) \in \mathbb{R}^{\lvert\mathcal{S}\rvert \times B}$ for a confocal setup $I(\mathcal{T}) \in \mathbb{R}^{\lvert\mathcal{L}\rvert \times \lvert\mathcal{S}\rvert \times B}$ for an exhaustive setup
$v(x, y)$	Binary visibility between points x and y
α	Light transport function
ω	Weighting function
b	A transient bin index
B	Number of transient bins
ϕ	Bin offset of the first bin
δ	Bin width
$\mathcal{T}$	A set of triangles $\mathcal{T} = \{t_1, \ldots, t_n\}$
t_i	A triangle $\in \mathbb{R}^9$ of three vertices $v_{i,0}$, $v_{i,1}$, $v_{i,2}$
$v_{i,j}$	The j-th vertex $\in \mathbb{R}^3$ of triangle i
a_i	Albedo values $\in \mathbb{R}^3$ of triangle i
s	A scan point on the visible wall
n_s	Surface normal at s
$\mathcal{S}$	The set of all scan points
o_s	The position of the scanning device
l	A laser point on the visible wall
n_l	Surface normal at l
$\mathcal{L}$	The set of all laser points
o_l	The position of the laser source
$\mathbf{1}_n$	A vector of ones of length n
J_f	Jacobian of a function f
$\otimes$	Kronecker product

Table 1. Definitions of symbols used

BRDF We chose to set the average vertex albedo as a constant term for lambertian reflection. Other differentiable BRDF functions could also be implemented.

$$a(a_i) = \frac{1}{3}(a_{i,0} + a_{i,1} + a_{i,2}) \tag{5}$$

$$\nabla_{a_i} a(a_i) = \frac{1}{3}\mathbf{1}_3 \tag{6}$$

1.2. Confocal Setup

We start by recapitulating the formulation of our rendering function I and its derivative before giving an explanation of the light transport function α, the weighting function ω, and their respective derivatives. For a derivation of how to arrive at this formulation of the rendering function and some intuition behind it we refer to the original work of Iseringhausen and Hullin [2]. The rendering function is defined as

$$I(\mathcal{T}) = \left(\sum_{i=1}^{n} v(s, c(t_i))\alpha(s, t_i)\omega(s, b, t_i) \right)_{s,b}. \tag{7}$$

For an arbitrary loss function $L(I)$ we compute gradients for each triangle t_i and for each triangle's albedo a_i through backpropagation as

$$\nabla_{t_i} L = \sum_{s \in \mathcal{S}} \sum_{b=0}^{B-1} \frac{\partial L}{\partial I_{s,b}} \nabla_{t_i} I_{s,b} \tag{8}$$

and

$$\nabla_{a_i} L = \sum_{s \in \mathcal{S}} \sum_{b=0}^{B-1} \frac{\partial L}{\partial I_{s,b}} \nabla_{a_i} I_{s,b}. \tag{9}$$

Note that we can combine both computations in a single function to reduce overhead. As they are conceptually similar we will only state formulas for the albedo gradient where they differ from the coordinate gradient. Most importantly, the weights ω do not depend on the albedo and hence $\nabla_{a_i}\omega$ will be zero. We reformulate this as

$$\nabla_{t_i} L = \sum_{s \in \mathcal{S}} v(s, t_i) \left(\nabla_{t_i} \alpha(s,t) \sum_{b=0}^{B-1} \frac{\partial L}{\partial I_{s,b}} \omega(s,b,t) + \alpha(s,t) \sum_{b=0}^{B-1} \frac{\partial L}{\partial I_{s,b}} \nabla_{t_i} \omega(s,b,t) \right). \tag{10}$$

This formulation is only correct for points where the visibility is constant within an ϵ-ball around t_i as the visibility function is discontinuous along edges. We opt to ignore gradients resulting from visibility changes to keep computational complexity low.

Light Transport Function Let us first state the full definition of the light transport function α to provide intuition of the individual terms before simplifying the expression for the implementation and the gradient computation:

$$\alpha(s, t_i) = a(t_i) \|n(t_i)\| \frac{1}{\|s - c(t_i)\|^4} \langle n_s, \frac{c(t_i) - s}{\|c(t_i) - s\|} \rangle^2 \langle \frac{n(t_i)}{\|n(t_i)\|}, \frac{s - c(t_i)}{\|s - c(t_i)\|} \rangle^2 \tag{11}$$

From left to right, the individual terms are the BRDF function, the area of the triangle, the inverse-square law between the wall and the centroid (and back), the cosine between the wall normal and the ray, and the cosine between the triangle normal an the ray. We can simplify the expression to

$$\alpha(s, t_i) = a(t_i) \frac{\langle n_s, c(t_i) - s \rangle^2 \langle n(t_i), c(t_i) - s \rangle^2}{\|n(t_i)\| \|c(t_i) - s\|^8}. \tag{12}$$

We compute the gradient of α with respect to the triangle using logarithmic derivatives as

$$\begin{aligned} \nabla_{t_i} \alpha(s, t_i) = \alpha(s, t_i) \Bigg(& \frac{\nabla_{t_i} a(t_i)}{a(t_i)} + \frac{\nabla_{t_i} \|n(t_i)\|^{-1}}{\|n(t_i)\|^{-1}} + \frac{\nabla_{t_i} \|c(t_i) - s\|^{-8}}{\|c(t_i) - s\|^{-8}} + \\ & \frac{\nabla_{t_i} \langle n_s, c(t_i) - s \rangle^2}{\langle n_s, c(t_i) - s \rangle^2} + \frac{\nabla_{t_i} \langle n(t_i), c(t_i) - s \rangle^2}{\langle n(t_i), c(t_i) - s \rangle^2} \Bigg), \end{aligned} \tag{13}$$

which yields

$$\begin{aligned} \nabla_{t_i} \alpha(s, t_i) = \alpha(s, t_i) \Bigg(& - J_n(t_i) \frac{n(t_i)}{\|n(t_i)\|^2} - \frac{8}{3\|c(t) - s\|^2} \mathbf{1}_3 \otimes (c(t_i) - s) + \\ & \frac{2}{3\langle n_s, c(t) - s \rangle} \mathbf{1}_3 \otimes n_s + \\ & \frac{2}{\langle n(t), c(t) - s \rangle} (J_n(t_i) \, (c(t) - s) + \frac{1}{3} \mathbf{1}_3 \otimes n(t_i)) \Bigg). \end{aligned} \tag{14}$$

The gradient with respect to the albedo values of the vertices can be computed as

$$\nabla_{a_i} \alpha(s, t_i) = \nabla_{a_i} a(t_i) \frac{\langle n_s, c(t_i) - s \rangle^2 \langle n(t_i), c(t_i) - s \rangle^2}{\|n(t_i)\| \|c(t_i) - s\|}. \tag{15}$$

Weighting Function To distribute the computed α on the affected transient bins, we compute the corresponding bin of each vertex using the total distance of the light traveled:

$$\theta(v_{i,j}) = (2\|v_{i,j} - s\|_2 + 2\|s - o_s\|_2 - \phi)/\delta \tag{16}$$

Note that the distance between the wall point and the sensor ($\|s-o_s\|_2$) is constant for all vertices. To keep the implementation flexible we provide options in the interface to ignore this term completely when rectified measurements are considered. The gradient of θ is

$$\nabla_{v_{i,j}}\theta(v_{i,j}) = \frac{2}{\delta\|v_{i,j}-s\|_2}(v_{i,j}-s) \tag{17}$$

In the following, we assume that $v_{i,0}$, $v_{i,1}$, and $v_{i,2}$ are sorted in order of ascending distance $\theta(\cdot)$. Outside the affected bins we set

$$\omega(s,b,t_i) = 0, \quad b < \lfloor\theta(v_{i,0})\rfloor \vee b > \lfloor\theta(v_{i,2})\rfloor. \tag{18}$$

This greatly simplifies the computation of both the forward and the backward pass, as $\theta(v_{i,0})$ and $\theta(v_{i,2})$ can be computed early on.

Let us first consider the special case $\lfloor\theta(v_{i,0})\rfloor = \lfloor\theta(v_{i,1})\rfloor = \lfloor\theta(v_{i,2})\rfloor$. Here, all light falls into a single bin and we can set

$$\omega(s, \lfloor\theta(v_{i,0})\rfloor, t) = 1 \tag{19}$$

and the gradient is a zero vector.

In the case where $\lfloor\theta(v_{i,0})\rfloor < \lfloor\theta(v_{i,2})\rfloor$, we use a trapezoidal filter for the weights as shown in Fig. 1. We consider the computation of weights as the area under the first triangle ω_1 and the second triangle ω_2 independently and set $\omega = \omega_1 + \omega_2$. Using this approach we can take advantage of the symmetry between both cases which lets us reuse the same function and its derivative. Note that only the weight of bin $\lfloor\theta(v_{i,1})\rfloor$ is composed of the area under both triangles, while for all other weights one term will be zero.

To construct the weights we define a central value

$$\omega_c(t_i) = \frac{2}{\theta(v_{i,2}) - \theta(v_{i,0})} \tag{20}$$

which is located at $\theta(v_{i,1})$. The corresponding gradient is

$$\nabla_{t_i}\omega_c(t_i) = -\frac{4}{\delta(\theta(v_{i,2})-\theta(v_{i,0}))^2}\begin{bmatrix} -\nabla_{v_{i,0}}\theta(v_{i,0}) \\ \begin{pmatrix}0\\0\\0\end{pmatrix} \\ \nabla_{v_{i,2}}\theta(v_{i,2}) \end{bmatrix}. \tag{21}$$

In the special case where either the first or the second triangle falls into a single bin, i.e. $\lfloor\theta(v_{i,0})\rfloor = \lfloor\theta(v_{i,1})\rfloor$ or $\lfloor\theta(v_{i,1})\rfloor = \lfloor\theta(v_{i,2})\rfloor$, we can trivially compute the weights as the full area under the triangles as

$$\omega_1(s,b,t_i) = \frac{1}{2}\omega_c(t_i)(\theta(v_{i,1}) - \theta(v_{i,0})), \quad \text{or} \tag{22}$$

$$\omega_2(s,b,t_i) = \frac{1}{2}\omega_c(t_i)(\theta(v_{i,2}) - \theta(v_{i,1})). \tag{23}$$

Otherwise we compute the weights using a function $\rho(x_1, y_1, x_2, y_2, b)$ that computes the area under the line defined by the points (x_1, y_1) and (x_2, y_2) between b and $b+1$. With this, we can set

$$\omega_1(s,b,t_i) = \rho(\theta(v_{i,0}), 0, \theta(v_{i,1}), \omega_c(t_i), b) \tag{24}$$

$$\omega_2(s,b,t_i) = \rho(\theta(v_{i,1}), \omega_c(t_i), \theta(v_{i,2}), 0, b) \tag{25}$$

For the definition of ρ we need to distinguish special cases for the first and the last bin. All equations are derived from the length of the base times the height at the center of the bin, which is interpolated from the line definition:

$$\rho(x_1,y_1,x_2,y_2,b) = \begin{cases} (b+1-x_1)(y_1+(\frac{1}{2}(x_1+b+1)-x_1)\frac{y_2-y_1}{x_2-x_1}) & ,b=\lfloor x_1\rfloor \\ y_1+(b+\frac{1}{2}-x_1)\frac{y_2-y_1}{x_2-x_1} & ,\lfloor x_1\rfloor<b<\lfloor x_2\rfloor \\ (x_2-b)(y_1+(\frac{1}{2}(b+x_2)-x_1)\frac{y_2-y_1}{x_2-x_1}) & ,b=\lfloor x_2\rfloor \end{cases} \tag{26}$$

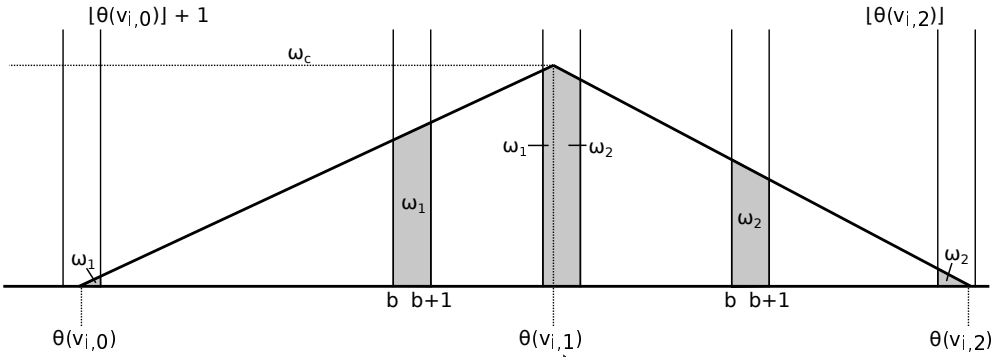

Figure 1. Visual representation of the computation of weights ω for various bins showing the intermediate and boundary cases

For sake of completeness, the gradients for each case are stated below. x_1, y_1, x_2, and y_2 need to be substituted as stated in Eq. (24) and Eq. (25) to arrive at the final result. To shorten the notation we write the gradient of $\delta(x_1, y_1, x_2, y_2) := \frac{y_2(t_i)-y_1(t_i)}{x_2(t_i)-x_1(t_i)}$ as

$$\nabla_{t_i}\delta(x_1,y_1,x_2,y_2)=\frac{(\nabla_{t_i}y_2(t_i)-\nabla_{t_i}y_1(t_i))(x_2(t_i)-x_1(t_i))-(y_2(t_i)-y_1(t_i))(\nabla_{t_i}x_2(t_i)-\nabla_{t_i}x_1(t_i))}{(x_2(t_i)-x_1(t_i))^2} \tag{27}$$

$b = \lfloor x_1 \rfloor$:

$$\begin{aligned}\nabla_{t_i}\rho(x_1, y_1, x_2, y_2, b) =& \nabla_{t_i}y_1(t_i)(b + 1 - x_1(t_i)) - \nabla_{t_i}x_1(t_i)y_1(t_i)\\ &- \nabla_{t_i}x_1(t_i)(b + 1 - x_1(t_i))\delta(x_1, y_1, x_2, y_2)\\ &+ \frac{1}{2}(b + 1 - x_1(t_i))^2\nabla_{t_i}\delta(x_1, y_1, x_2, y_2)\end{aligned} \tag{28}$$

$\lfloor x_1 \rfloor < b < \lfloor x_2 \rfloor$:

$$\begin{aligned}\nabla_{t_i}\rho(x_1, y_1, x_2, y_2, b) =& \nabla_{t_i}y_1(t_i) - \nabla_{t_i}x_1(t_i)\delta(x_1, y_1, x_2, y_2)\\ &+ \big(b + \frac{1}{2} - x_1(t_i)\big)\nabla_{t_i}\delta(x_1, y_1, x_2, y_2)\end{aligned} \tag{29}$$

$b = \lfloor x_2 \rfloor$:

$$\begin{aligned}\nabla_{t_i}\rho(x_1, y_1, x_2, y_2, b) =& \nabla_{t_i}x_2(t_i)\left((y_1(t_i) + (\frac{1}{2}(b + x_2(t_i)) - x_1(t_i))\delta(x_1, y_1, x_2, y_2)\right)\\ &+ (x_2(t_i) - b)\Big(\\ &\quad \nabla_{t_i}y_1(t_i)\\ &\quad + (\frac{1}{2}\nabla_{t_i}x_2(t_i) - \nabla_{t_i}x_1(t_i))\delta(x_1, y_1, x_2, y_2)\\ &\quad + (\frac{1}{2}(b + x_2(t_i)) - x_1(t_i))\nabla_{t_i}\delta(x_1, y_1, x_2, y_2)\\ &\Big)\end{aligned} \tag{30}$$

1.3. Non-Confocal Setup

The equations of the exhaustive scanning setup conceptually follow those of the confocal setup. Therefore, we will only detail significant differences to the derivations given before.

The rendering function takes the points l into account and is defined as

$$I(\mathcal{T}) = \left(\sum_{i=1}^{n} v(l, c(t_i))v(s, c(t_i))\alpha(l, s, t_i)\omega(l, s, b, t_i)\right)_{l,s,b}. \tag{31}$$

The gradient equations simply replace all summations over the scan points $\mathcal{S}$ with summations over both $\mathcal{S}$ and $\mathcal{L}$.

Light Transport Function We adapt the light transport function to the new paths from the laser to l, to the triangle t_i, to the point s and to the detector:

$$\begin{aligned}\alpha(l,s,t_i) =& a(t_i)\cdot\|n(t_i)\|\cdot\frac{1}{\|l-c(t_i)\|^2}\cdot\frac{1}{\|s-c(t_i)\|^2}\\ &\cdot|\langle n_l,\frac{c(t_i)-l}{\|c(t_i)-l\|}\rangle|\cdot|\langle n_s,\frac{c(t_i)-s}{\|c(t_i)-s\|}\rangle|\\ &\cdot|\langle\frac{n(t_i)}{\|n(t_i)\|},\frac{l-c(t_i)}{\|l-c(t_i)\|}\rangle|\cdot|\langle\frac{n(t_i)}{\|n(t_i)\|},\frac{s-c(t_i)}{\|s-c(t_i)\|}\rangle|\end{aligned}\tag{32}$$

We perform a similar simplification and rewrite α as

$$\alpha(s,t_i)=a(t_i)\frac{|\langle n_l,c(t_i)-l\rangle|\cdot|\langle n_s,c(t_i)-s\rangle|\cdot|\langle n(t_i),c(t_i)-l\rangle|\cdot|\langle n(t_i),c(t_i)-s\rangle|}{\|n(t_i)\|\|c(t_i)-l\|^4\|c(t_i)-s\|^4}.\tag{33}$$

Differentiating α using logarithmic derivatives

$$\begin{aligned}\nabla_{t_i}\alpha(l,s,t_i)=\alpha(l,s,t_i)\Bigg(&\frac{\nabla_{t_i}a(t_i)}{a(t_i)}+\frac{\nabla_{t_i}\|n(t_i)\|^{-1}}{\|n(t_i)\|^{-1}}+\\ &\frac{\nabla_{t_i}\|c(t_i)-l\|^{-4}}{\|c(t_i)-l\|^{-4}}+\frac{\nabla_{t_i}\|c(t_i)-s\|^{-4}}{\|c(t_i)-s\|^{-4}}+\\ &\frac{\nabla_{t_i}|\langle n_l,c(t_i)-l\rangle|}{|\langle n_l,c(t_i)-l\rangle|}+\frac{\nabla_{t_i}|\langle n_s,c(t_i)-s\rangle|}{|\langle n_s,c(t_i)-s\rangle|}+\\ &\frac{\nabla_{t_i}|\langle n(t_i),c(t_i)-l\rangle|}{|\langle n(t_i),c(t_i)-l\rangle|}+\frac{\nabla_{t_i}|\langle n(t_i),c(t_i)-s\rangle|}{|\langle n(t_i),c(t_i)-s\rangle|}\Bigg)\end{aligned}\tag{34}$$

yields

$$\begin{aligned}\nabla_{t_i}\alpha(l,s,t_i)=\alpha(l,s,t_i)\Bigg(&-J_n(t_i)\frac{n(t_i)}{\|n(t_i)\|^2}-\\ &\frac{4}{3\|c(t_i)-l\|^2}\mathbf{1}_3\otimes(c(t_i)-l)-\\ &\frac{4}{3\|c(t_i)-s\|^2}\mathbf{1}_3\otimes(c(t_i)-s)+\\ &\frac{1}{3\langle n_s,c(t_i)-l\rangle}\mathbf{1}_3\otimes n_l+\\ &\frac{1}{3\langle n_s,c(t_i)-s\rangle}\mathbf{1}_3\otimes n_s+\\ &\frac{1}{\langle n(t_i),c(t_i)-l\rangle}(J_n(t_i)\,(c(t_i)-l)+\frac{1}{3}\mathbf{1}_3\otimes n(t_i))+\\ &\frac{1}{\langle n(t_i),c(t_i)-s\rangle}(J_n(t_i)\,(c(t_i)-s)+\frac{1}{3}\mathbf{1}_3\otimes n(t_i))\Bigg).\end{aligned}\tag{35}$$

Weighting Function The equations for the weighting function are mostly unchanged. The only difference is the computation of the transient bins for each vertex, which is

$$\theta(v_{i,j})=(\|v_{i,j}-l\|_2+\|v_{i,j}-s\|_2+\|l-o_l\|_2+\|s-o_s\|_2-\phi)/\delta\tag{36}$$

and its gradient

$$\nabla_{v_{i,j}}\theta(v_{i,j})=\frac{1}{\delta\|v_{i,j}-l\|_2}(v_{i,j}-l)+\frac{1}{\delta\|v_{i,j}-s\|_2}(v_{i,j}-s).\tag{37}$$

Again, we enable our implementation to ignore $\|l-o_l\|_2$, $\|s-o_s\|_2$, or both.

2. Algorithms

Here, we give simplified pseudo code listings for the reconstruction algorithms using radial basis function (Algorithm 1) and depth map (Algorithm 2) representations. Note that all functions except CHECKDELETE in Algorithm 1 also check if the loss has been reduced and otherwise return the input b. We use Adam [3] for all optimizations. In Algorithm 1 we start with a learning rate of 0.001 and reduce the learning rate whenever the loss increases, whereas in Algorithm 2 we follow a learning rate schedule similar to the one proposed by Smith and Topin [6].

Algorithm 1 Reconstruction using radial basis functions

function RECONSTRUCT(transient, resolution, n, subdivide)
  $b \leftarrow [\,]$
  for all $i \in \{0, ..., n-1\}$ **do**
    if $i \equiv 0 \mod 5$ or $|b| = 0$ **then**
      $b \leftarrow$ ADDVOLUME(b, transient, resolution)
    else if $i \equiv 1 \mod 5$ **then**
      $b \leftarrow$ ADDSURFACE(b, transient, resolution)
    else if $i \equiv 2 \mod 5$ **then**
      $b \leftarrow$ SPLIT(b, transient, resolution)
    else if $i \equiv 3 \mod 5$ **then**
      $b \leftarrow$ CHECKDELETE(b, transient, resolution)
    else if $i \equiv 4 \mod 5$ **then**
      $b \leftarrow$ REFINE(b, transient, resolution)
    end if
    if $i \in$ subdivide **then**
      resolution $\leftarrow 2 \cdot$ resolution
    end if
  end for
  return b
end function

function ADDVOLUME(b, transient, resolution)
  trasientTest $\leftarrow$ RENDER(b, resolution)
  transientError $\leftarrow \max(0,$ transient $-$ transientTest$)$
  volumeError $\leftarrow$ BACKPROJECT(transientError, resolution)
  samples $\leftarrow$ MULTINOMIAL(volumeError, 5)
  $b \leftarrow b \bigcup$ samples
  $b \leftarrow$ OPTIMIZE(b, transient, resolution)
  return b
end function

function ADDSURFACE(b, transient, resolution)
  loss $\leftarrow$ RENDER_LOSS(b, resolution, transient)
  vertsGrad $\leftarrow$ BACKPROPAGATE_TO_VERTS(loss)
  samples $\leftarrow$ MULTINOMIAL(vertsGrad, 5)
  $b \leftarrow b \bigcup$ samples
  $b \leftarrow$ OPTIMIZE(b, transient, resolution)
  return b
end function

function SPLIT(b, transient, resolution)
    samples $\leftarrow$ MULTINOMIAL(b, 30)
    $b \leftarrow b \setminus$ samples
    samples.sigma $\leftarrow 0.75 \cdot$ samples.sigma
    offset $\leftarrow$ RANDOM( )
    $b \leftarrow b \bigcup$ (samples + offset) $\bigcup$ (samples $-$ offset)
    $b \leftarrow$ OPTIMIZE(b, transient, resolution)
    return b
end function

function CHECKDELETE(b, transient, resolution)
    loss $\leftarrow$ RENDER_LOSS(b, resolution, transient)
    for all $b_t \in b$ **do**
        lossTest $\leftarrow$ RENDER_LOSS($b \setminus b_t$, transient)
        if lossTest $\leq$ loss $\cdot$ deleteFactor **then**
            loss $\leftarrow$ lossTest
            $b \leftarrow b \setminus b_t$
        end if
    end for
    return b
end function

function REFINE(b, transient, resolution)
    $b \leftarrow$ OPTIMIZE(b, transient, resolution)
    return b
end function

Algorithm 2 Reconstruction using a depth map

function RECONSTRUCT(transient, resolution, n, subdivide)
    depth $\leftarrow$ INIT_DEPTH(resolution)
    color $\leftarrow$ INIT_COLOR(resolution)
    for all $i \in \{0, ..., n-1\}$ **do**
        if $i \in$ subdivide **then**
            depth $\leftarrow$ SUBDIVIDE(depth)
            color $\leftarrow$ SUBDIVIDE(color)
        end if
        // Render depth map and compute losses
        transientRendered $\leftarrow$ RENDER(depth, color)
        $L_{\text{data}} \leftarrow \|\text{transientRendered} - \text{transient}\|_2^2$
        $TV_{\text{depth}} \leftarrow$ TV(depth)
        $TV_{\text{color}} \leftarrow$ TV(color)
        $L \leftarrow L_{\text{data}} + \lambda_d TV_{\text{depth}} + \lambda_c TV_{\text{color}}$
        // Apply gradient descent step and clamp parameters
        grad $\leftarrow$ BACKWARD(L)
        depth, color $\leftarrow$ APPLY_GRAD(depth, color, grad)
        depth $\leftarrow \min(\max(\text{depth}, 0), 1)$
        color $\leftarrow \min(\max(\text{color}, 0), 1)$
    end for
    return depth, color
end function

3. Experiments and Timings

Scene	Measurement Size	Target	Runtime	Source
Zaragoza Bunny	$512 \times 64 \times 64$	Gaussian Rbf	4 m, 51 s	[1]
	$512 \times 64 \times 64$	Depth Map & Color	1 m, 34 s	
Sinogram	512×360	Gaussian Rbf	2 m, 25 s	
Mannequin	$512 \times 224 \times 59$	Gaussian Rbf	11 m, 11 s	[7]
Mannequin Synthetic	$512 \times 224 \times 59$	Gaussian Rbf	7 m, 48 s	Ours
Spot (Cow)	$512 \times 64 \times 64$	Gaussian Rbf	10 m, 27 s	Ours
Statue (180 min)	$512 \times 512 \times 512$	Depth Map & Color	2 m, 21 s	[4]
	$512 \times 128 \times 128$	Depth Map & Color	2 m, 21 s	
	$512 \times 64 \times 64$	Depth Map & Color	2 m, 35 s	
	$512 \times 32 \times 32$	Depth Map & Color	39 s	
Diffuse S	$256 \times 32 \times 32$	Depth Map & Color	2 m, 29 s	[5]
Flat-field	$256 \times 32 \times 32$	Depth Map & Color	2 m, 32 s	
Bunny	$256 \times 32 \times 32$	Position & Rotation	0.97 s	Ours
Two Armadillos	$256 \times 64 \times 32$	32×2 Pos. & Rot.	2 m, 16 s	Ours
Self-Supervised Training	$(32\times)256 \times 16 \times 16$	Network Parameters	67ms/batch	Ours

Table 2. All experiments and runtimes. Runtimes are measured on a desktop computer with a NVIDIA GeForce RTX 2080 Ti GPU with 11 GB VRAM. The VRAM needed to reconstruct a $512 \times 64 \times 64$ transient is about 1.4 GB.

4. Additional Results

Bunny The optimization of a transformation of a bunny is shown in Fig. 2. Even though the initial position and rotation are quite different from the target, the optimization converges quickly towards the true position.

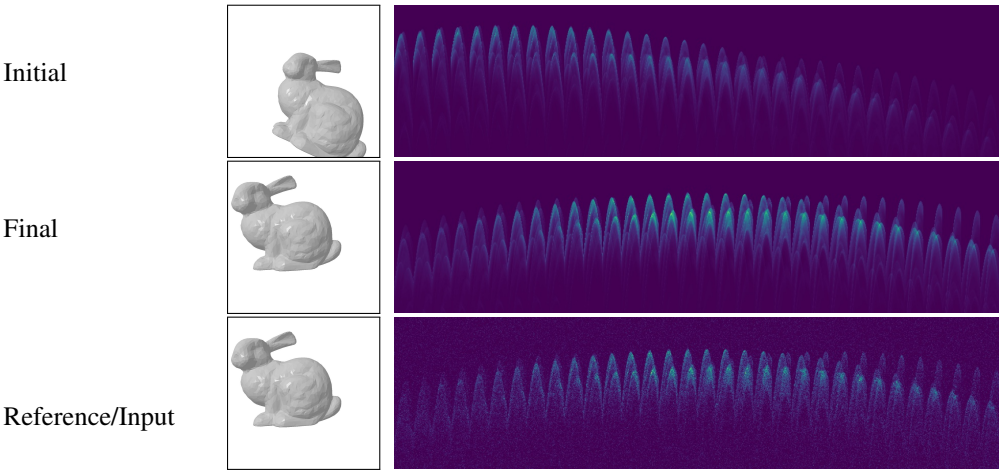

Figure 2. Optimizing position and rotation of a known object with 4968 triangles takes 0.97 seconds for a confocal measurement with a 32×32 resolution and 256 temporal bins

Statue Intermediate steps of the reconstruction of the statue are shown in Fig. 3. This example uses a 32×32 subset of the 180 min. measurement. We double the resolution of the depth map after 300 and 800 iterations.

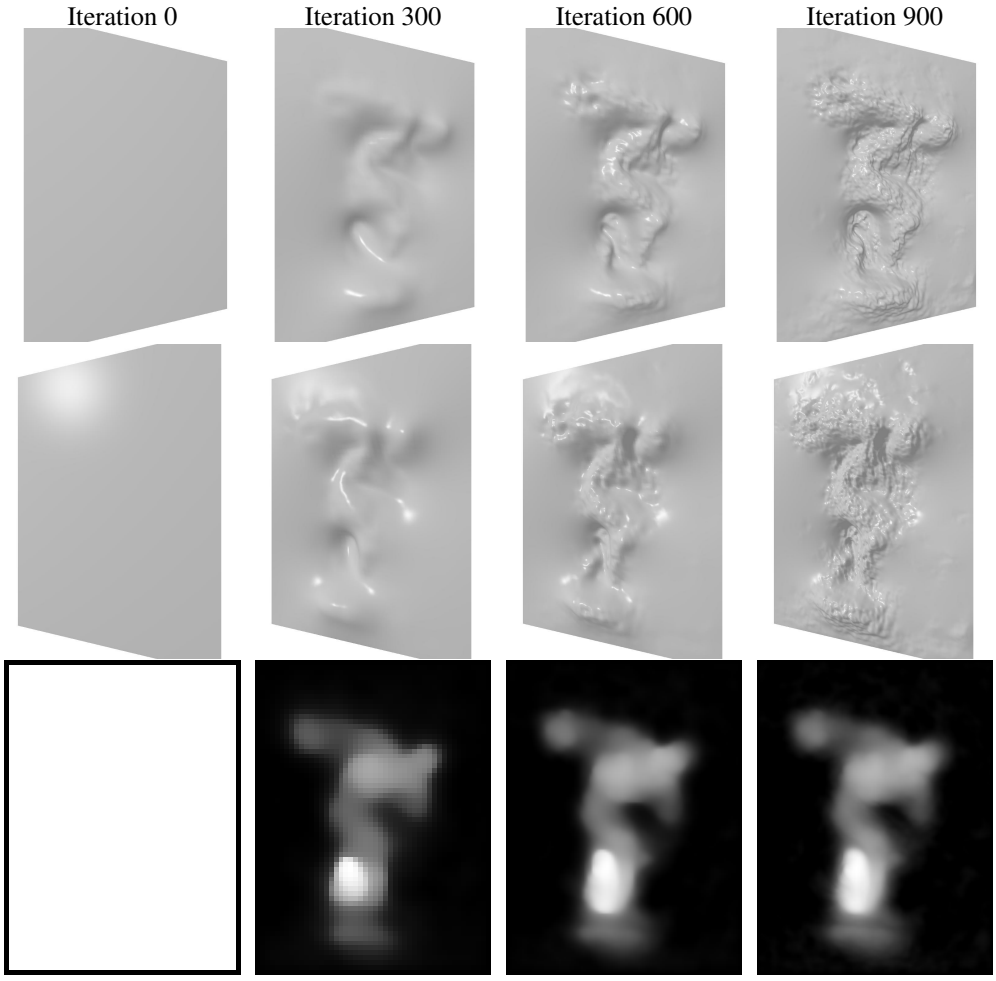

Figure 3. Optimization of the *Statue* measurement. We show the evolution of the shape (top and center row) and albedo (last row) separately for a better visualization. The resolution of the depth and albedo map is listed below the images

References

[1] Miguel Galindo, Julio Marco, Matthew O'Toole, Gordon Wetzstein, Diego Gutierrez, and Adrian Jarabo. A dataset for benchmarking time-resolved non-line-of-sight imaging, 2019.

[2] Julian Iseringhausen and Matthias B Hullin. Non-line-of-sight reconstruction using efficient transient rendering. *ACM Transactions on Graphics (TOG)*, 39(1):1–14, 2020.

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] David B Lindell, Gordon Wetzstein, and Matthew O'Toole. Wave-based non-line-of-sight imaging using fast fk migration. *ACM Transactions on Graphics (TOG)*, 38(4):1–13, 2019.

[5] Matthew O'Toole, David B Lindell, and Gordon Wetzstein. Confocal non-line-of-sight imaging based on the light-cone transform. *Nature*, 555(7696):338–341, 2018.

[6] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[7] Andreas Velten, Thomas Willwacher, Otkrist Gupta, Ashok Veeraraghavan, Moungi G Bawendi, and Ramesh Raskar. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nature communications*, 3(1):1–8, 2012.