SIGGRAPH ASIA 2024 TOKYO 東京

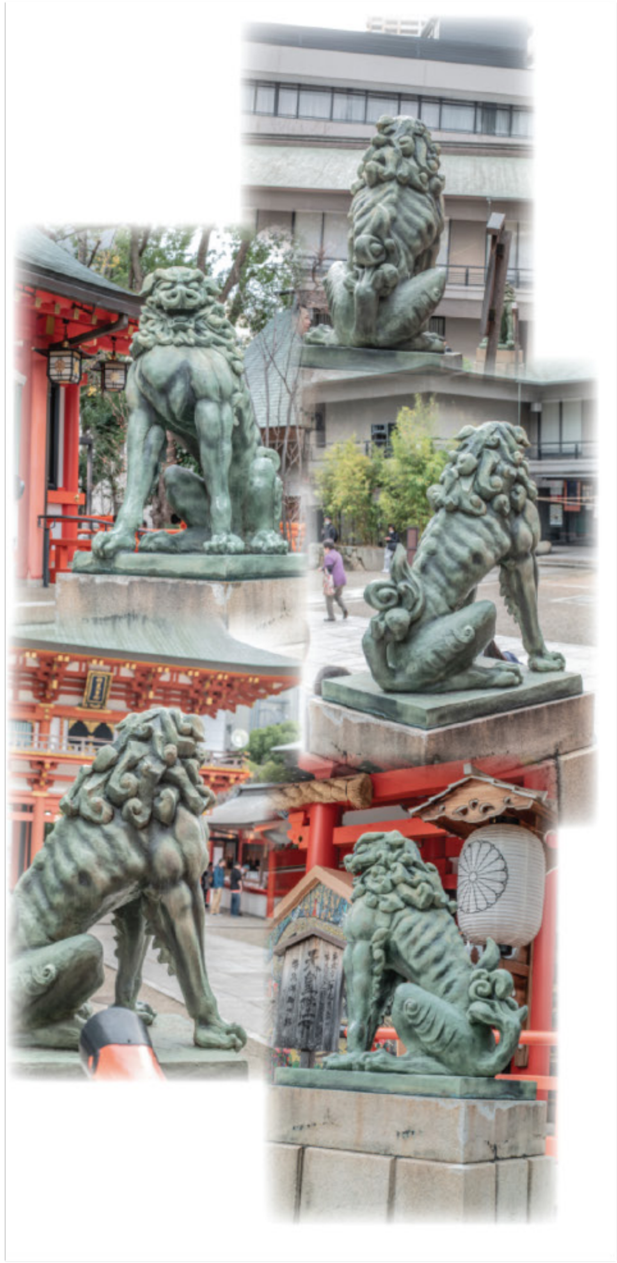
Conference | 3–6 December 2024
Exhibition | 4–6 December 2024
Venue | Tokyo International Forum, Japan

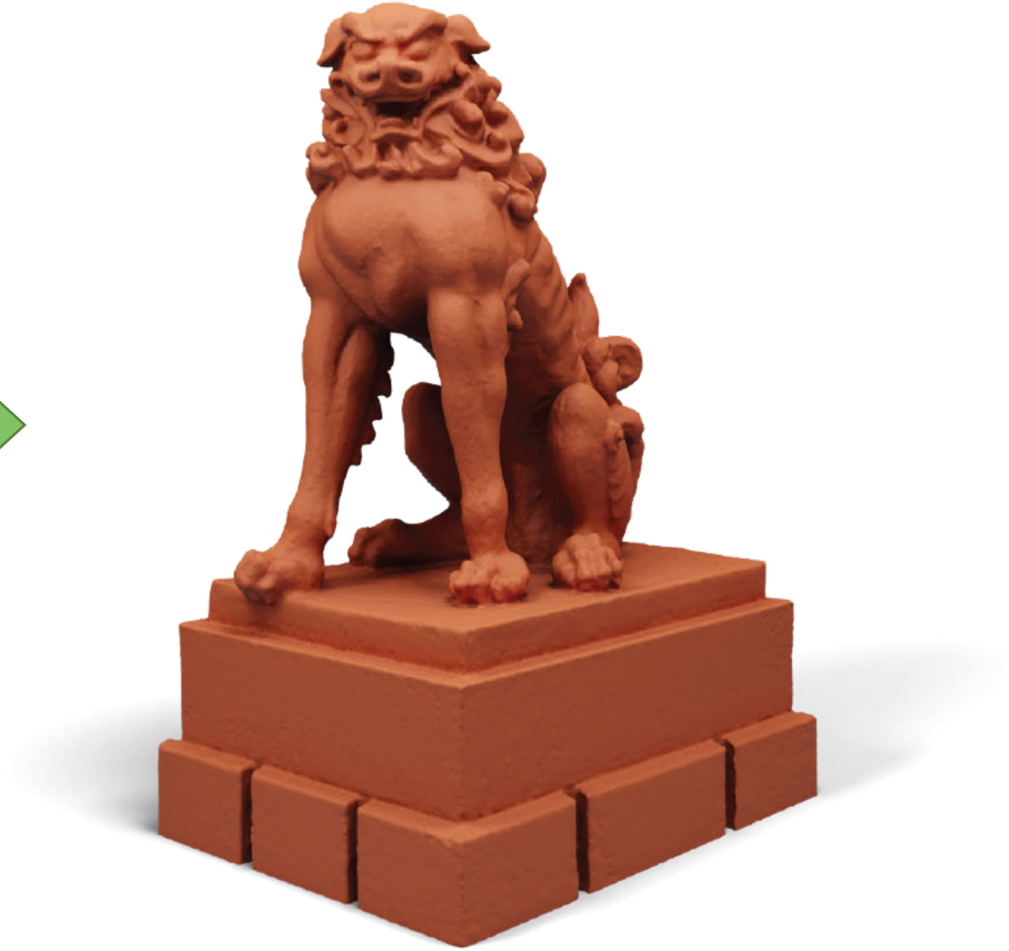
3D Reconstruction with Fast Dipole Sums

Hanyu Chen, Bailey Miller, Ioannis Gkioulekas

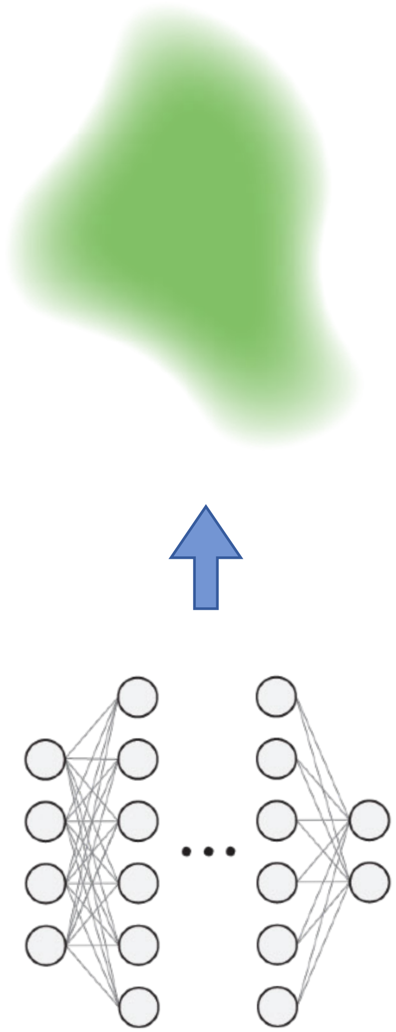
Sponsored by acm

Organized by koelnmesse

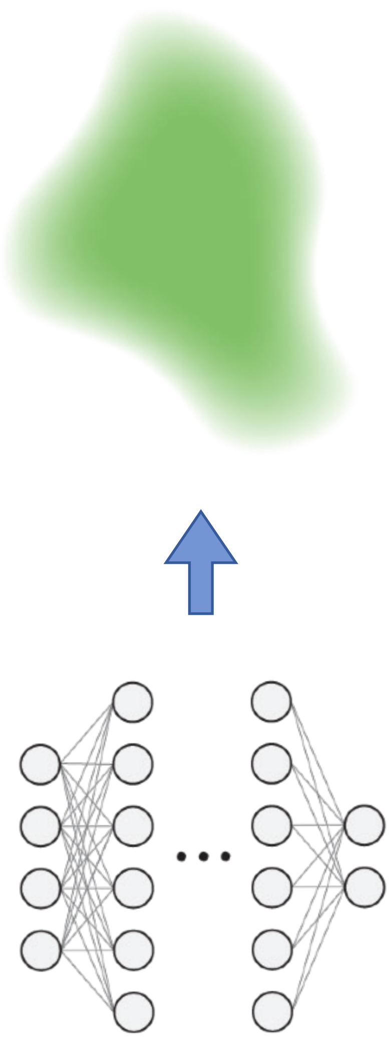




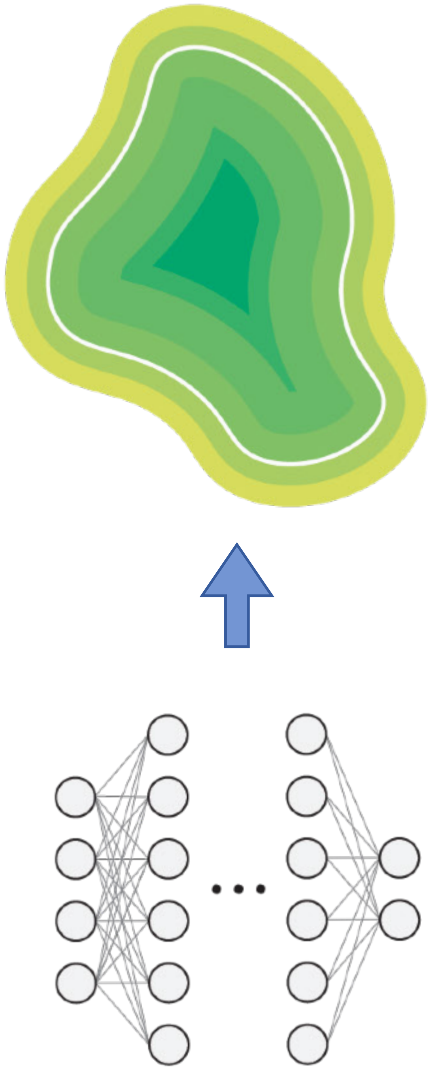
neural
radiance fields



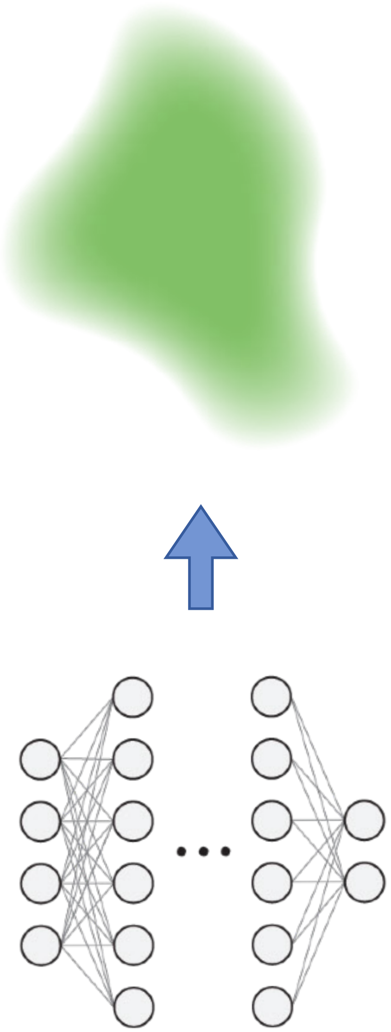
neural
radiance fields



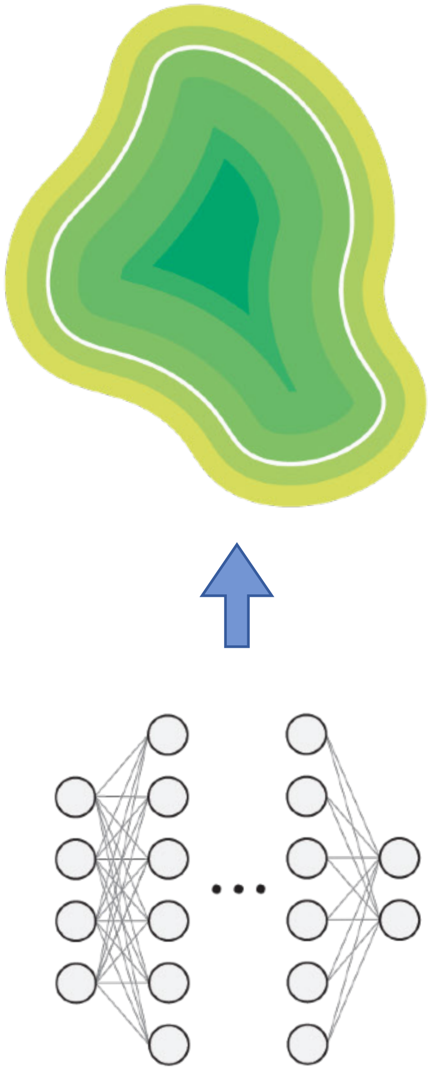
neural
implicit surfaces



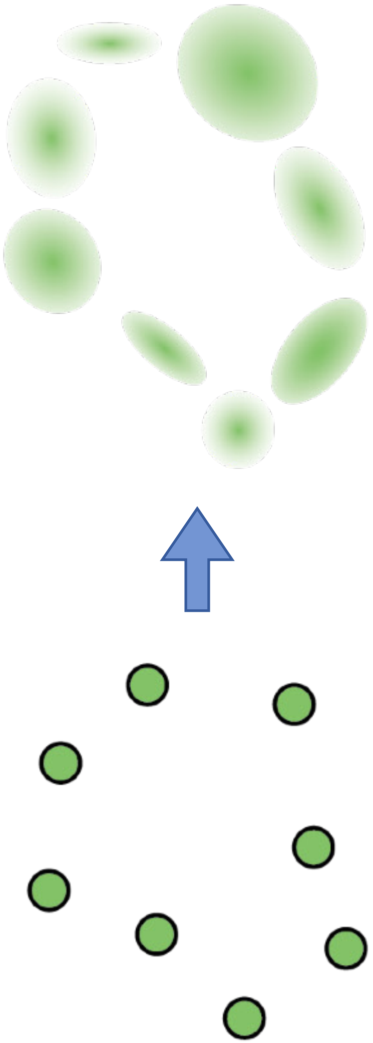
neural
radiance fields



neural
implicit surfaces



Gaussian
splatting

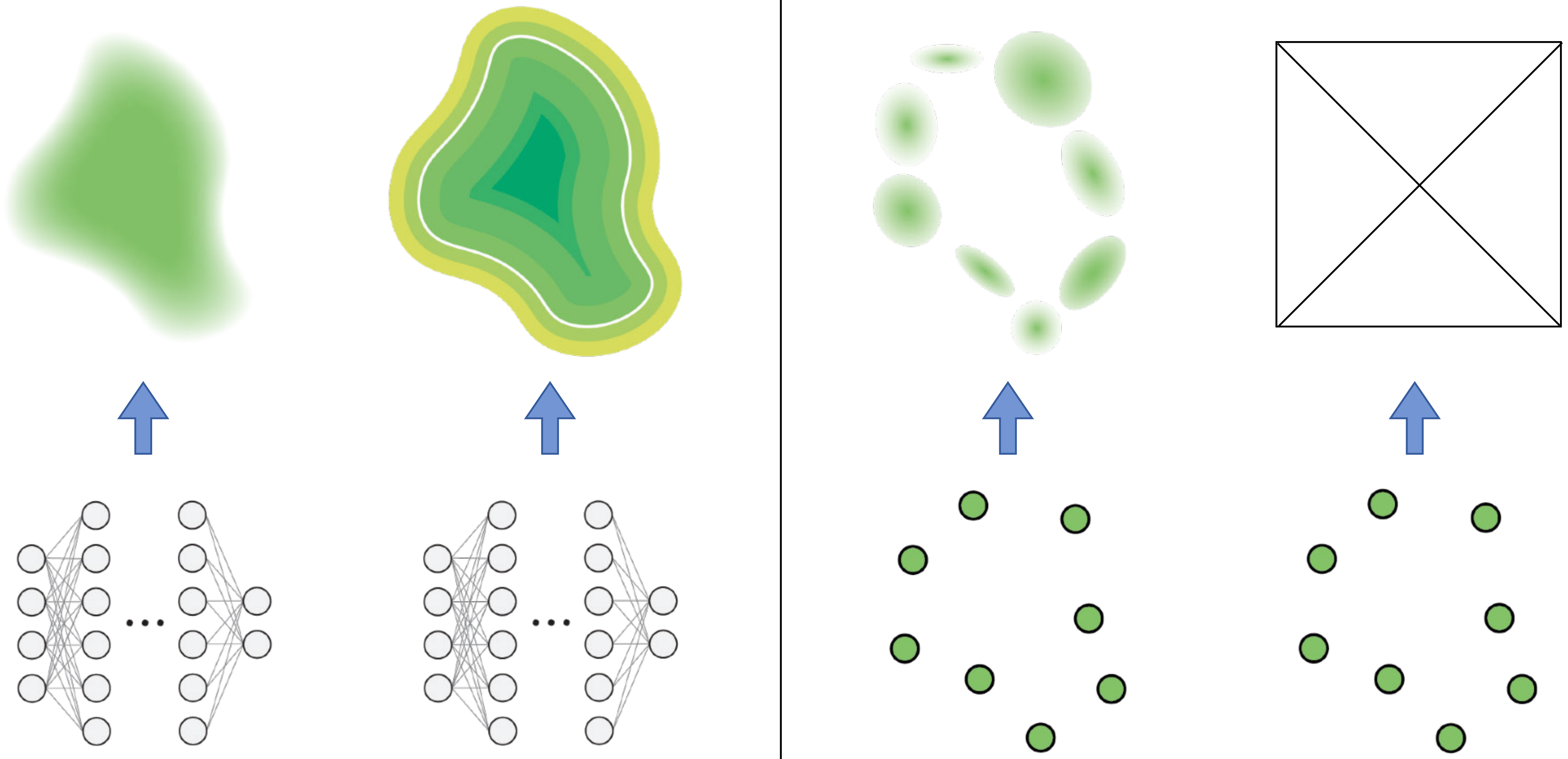


neural radiance fields

neural implicit surfaces

Gaussian splatting

?



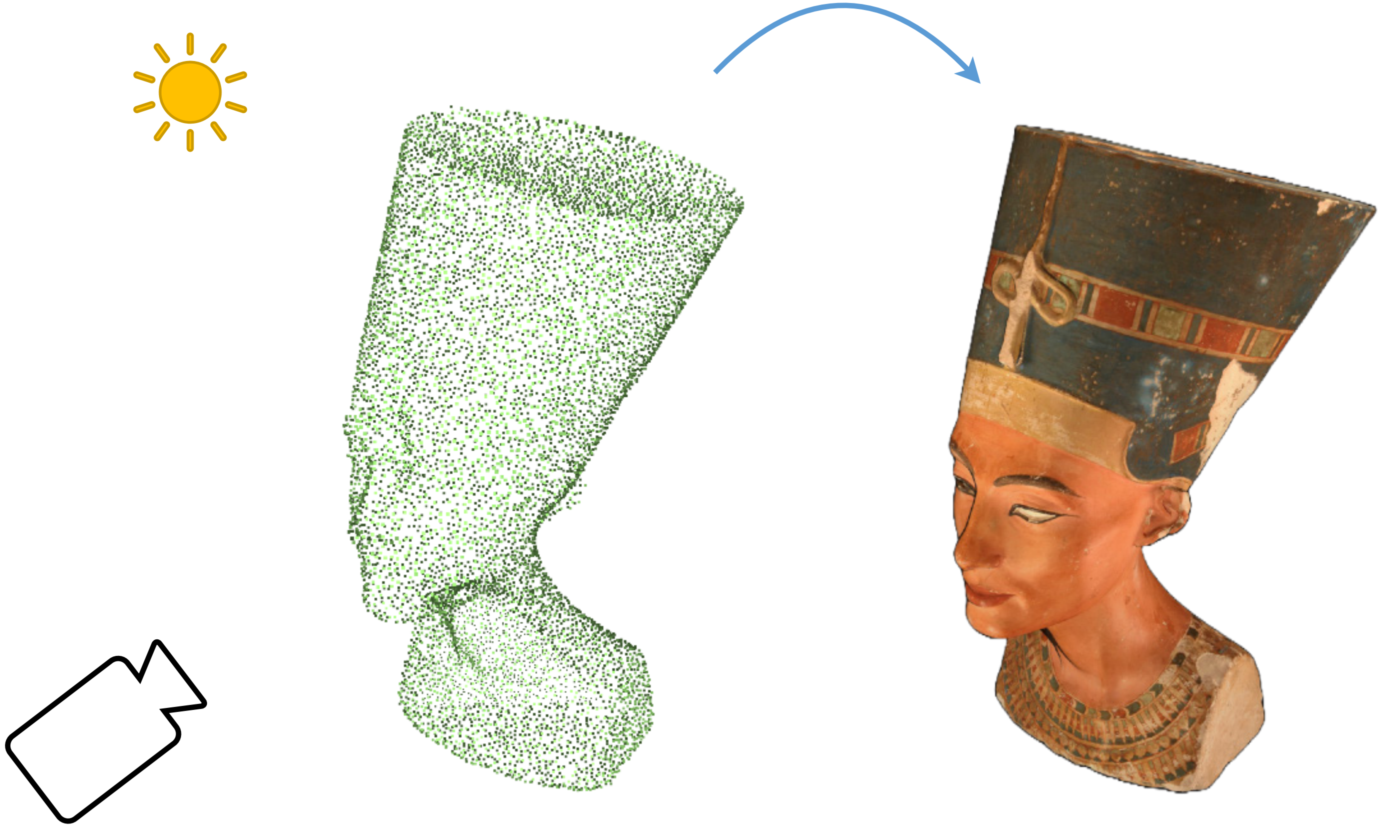
surface representation

desired properties:

surface representation

desired properties:

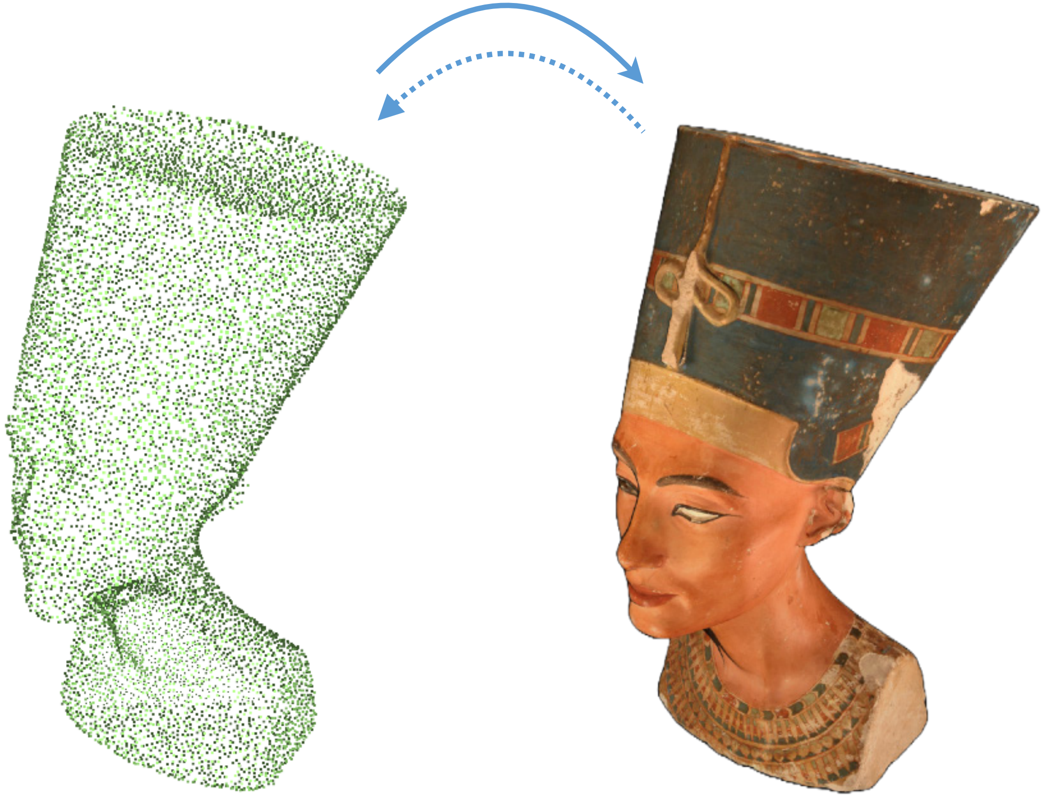
- efficient



surface representation

desired properties:

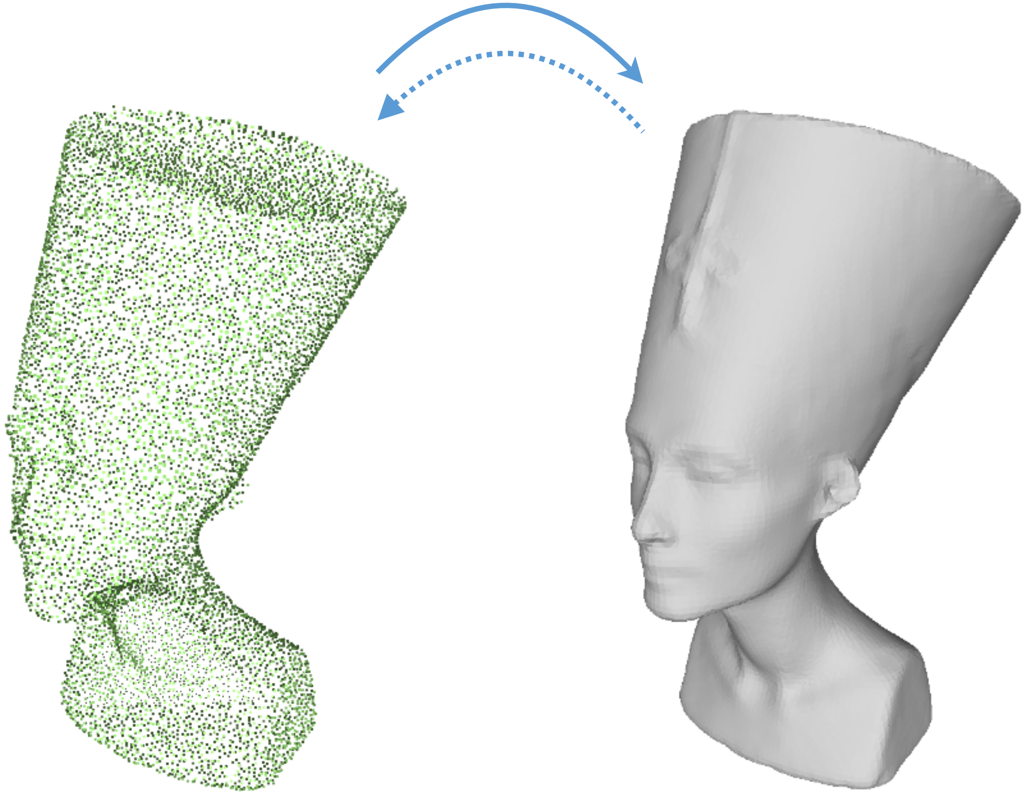
- efficient
- differentiable



surface representation

desired properties:

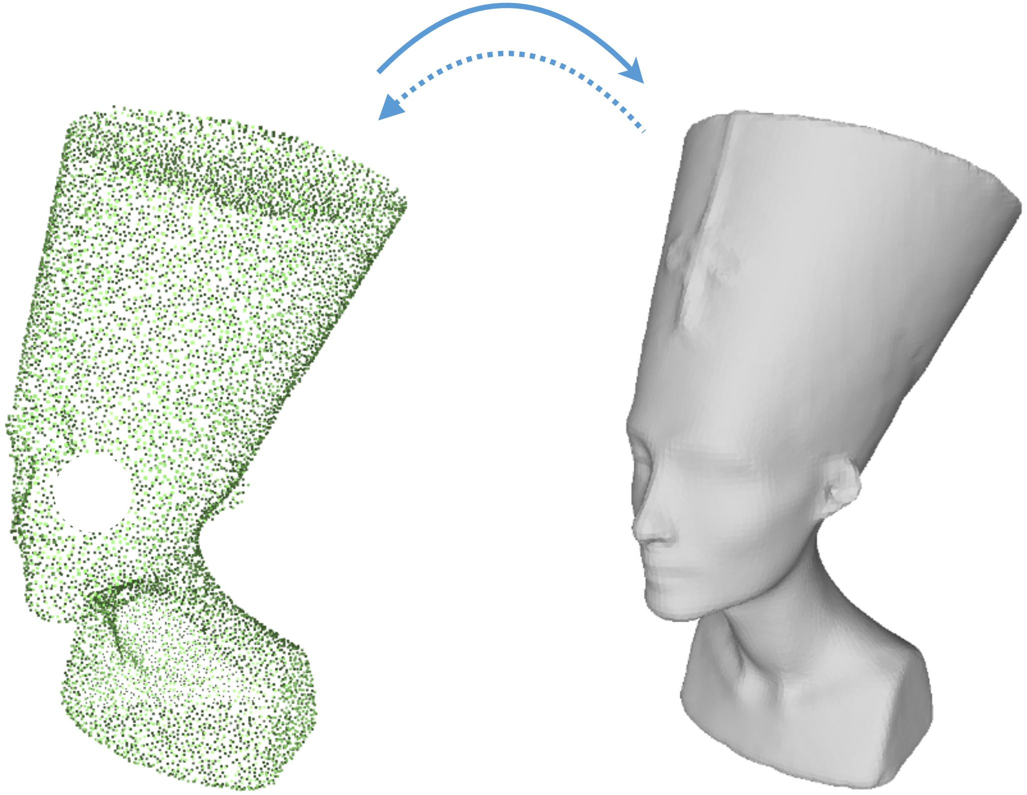
- efficient
- differentiable
- geometric regularity



surface representation

desired properties:

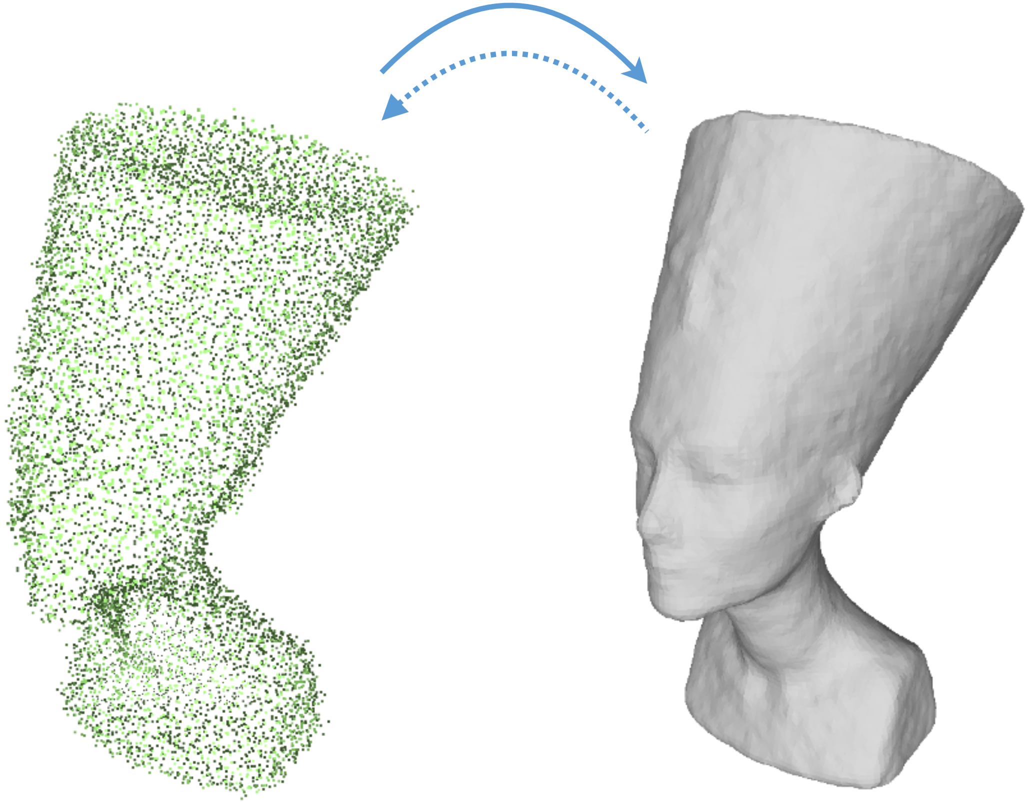
- efficient
- differentiable
- geometric regularity



surface representation

desired properties:

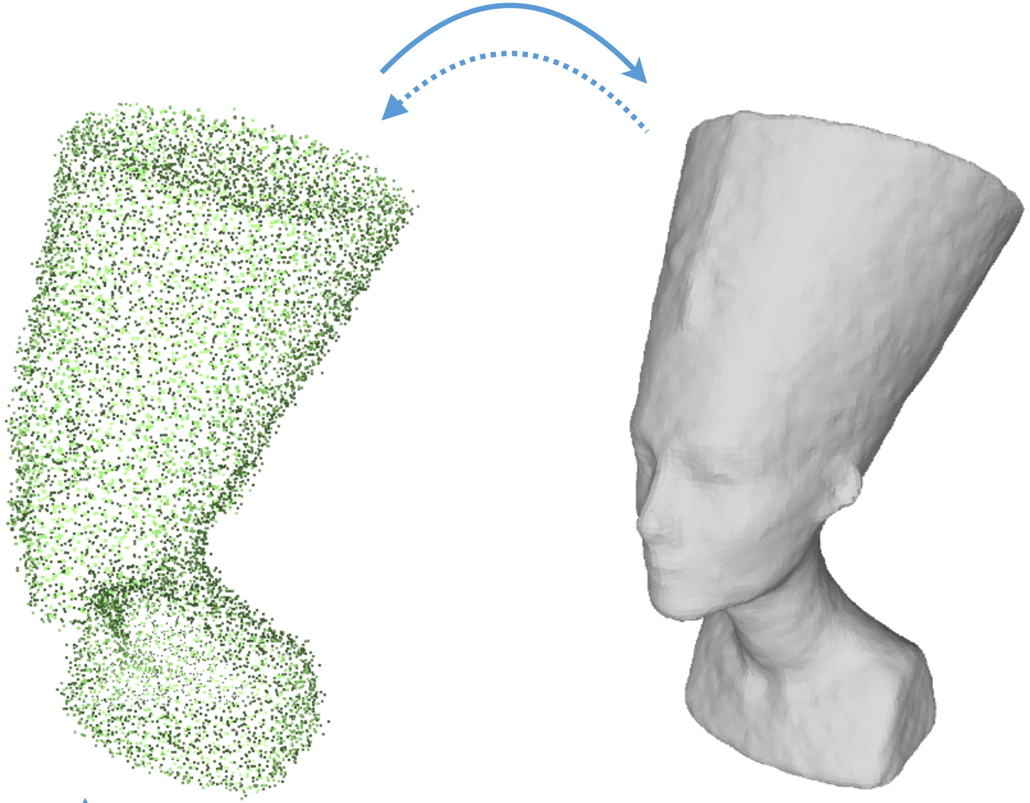
- efficient
- differentiable
- geometric regularity



surface representation

desired properties:

- efficient
- differentiable
- geometric regularity
- easy initialization



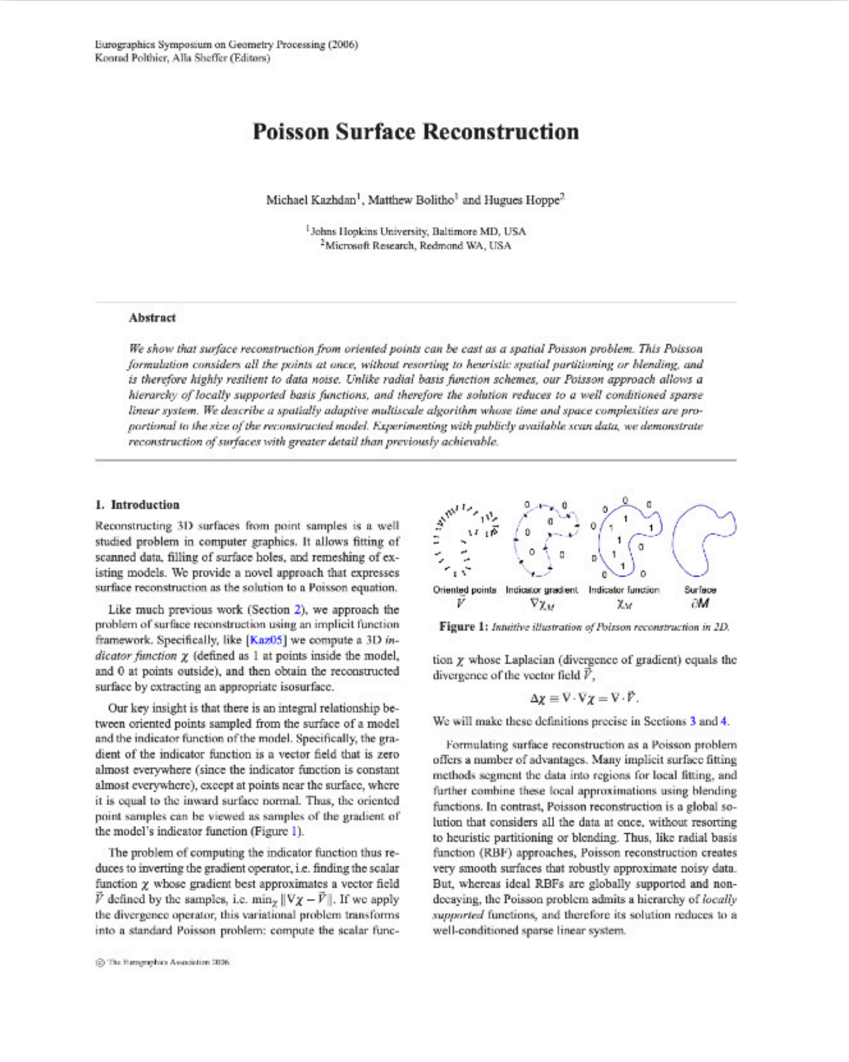
SfM & MVS (e.g. Colmap)

Poisson surface reconstruction

desired properties:

- efficient
- differentiable
- geometric regularity
- easy initialization

Kazhdan et al., “Poisson Surface Reconstruction”, Eurographics 2006
Peng et al. "Shape as points: A differentiable poisson solver”, NeurIPS 2021



Eurographics Symposium on Geometry Processing (2006)
Konrad Polthier, Alla Sheffer (Editors)

Poisson Surface Reconstruction

Michael Kazhdan[1], Matthew Bolitho[1] and Hugues Hoppe[2]

[1]Johns Hopkins University, Baltimore MD, USA
[2]Microsoft Research, Redmond WA, USA

Abstract

We show that surface reconstruction from oriented points can be cast as a spatial Poisson problem. This Poisson formulation considers all the points at once, without resorting to heuristic spatial partitioning or blending, and is therefore highly resilient to data noise. Unlike radial basis function schemes, our Poisson approach allows a hierarchy of locally supported basis functions, and therefore the solution reduces to a well conditioned sparse linear system. We describe a spatially adaptive multiscale algorithm whose time and space complexities are proportional to the size of the reconstructed model. Experimenting with publicly available scan data, we demonstrate reconstruction of surfaces with greater detail than previously achievable.

1. Introduction

Reconstructing 3D surfaces from point samples is a well studied problem in computer graphics. It allows fitting of scanned data, filling of surface holes, and remeshing of existing models. We provide a novel approach that expresses surface reconstruction as the solution to a Poisson equation.

Like much previous work (Section 2), we approach the problem of surface reconstruction using an implicit function framework. Specifically, like [Kaz05] we compute a 3D *indicator function* χ (defined as 1 at points inside the model, and 0 at points outside), and then obtain the reconstructed surface by extracting an appropriate isosurface.

Our key insight is that there is an integral relationship between oriented points sampled from the surface of a model and the indicator function of the model. Specifically, the gradient of the indicator function is a vector field that is zero almost everywhere (since the indicator function is constant almost everywhere), except at points near the surface, where it is equal to the inward surface normal. Thus, the oriented point samples can be viewed as samples of the gradient of the model's indicator function (Figure 1).

The problem of computing the indicator function thus reduces to inverting the gradient operator, i.e. finding the scalar function χ whose gradient best approximates a vector field $\vec{V}$ defined by the samples, i.e. $\min_\chi \|\nabla\chi - \vec{V}\|$. If we apply the divergence operator, this variational problem transforms into a standard Poisson problem: compute the scalar function χ whose Laplacian (divergence of gradient) equals the divergence of the vector field $\vec{V}$,

$$\Delta\chi \equiv \nabla\cdot\nabla\chi = \nabla\cdot\vec{V}.$$

We will make these definitions precise in Sections 3 and 4.

Figure 1: *Intuitive illustration of Poisson reconstruction in 2D.*

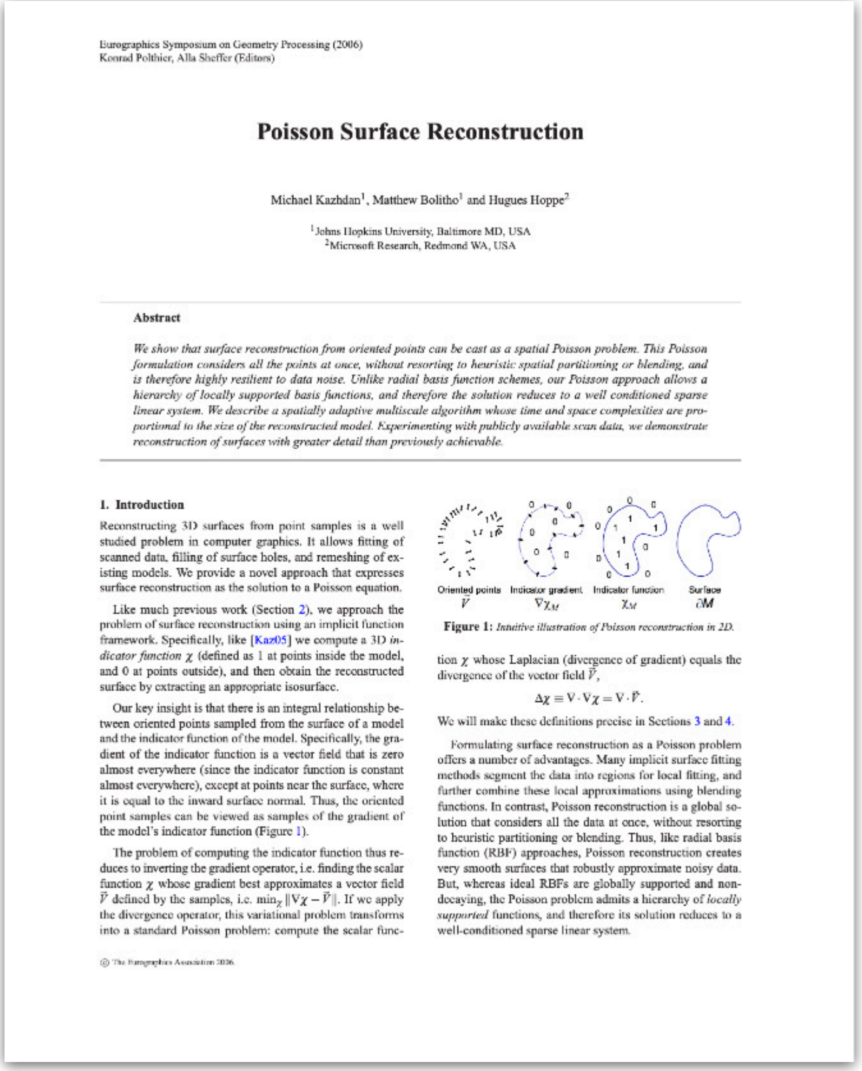
Formulating surface reconstruction as a Poisson problem offers a number of advantages. Many implicit surface fitting methods segment the data into regions for local fitting, and further combine these local approximations using blending functions. In contrast, Poisson reconstruction is a global solution that considers all the data at once, without resorting to heuristic partitioning or blending. Thus, like radial basis function (RBF) approaches, Poisson reconstruction creates very smooth surfaces that robustly approximate noisy data. But, whereas ideal RBFs are globally supported and non-decaying, the Poisson problem admits a hierarchy of *locally supported* functions, and therefore its solution reduces to a well-conditioned sparse linear system.

Poisson surface reconstruction

desired properties:

- efficient ✗
- differentiable ?
- geometric regularity ✓
- easy initialization ✓

Kazhdan et al., “Poisson Surface Reconstruction”, Eurographics 2006
Peng et al. "Shape as points: A differentiable poisson solver", NeurIPS 2021



Eurographics Symposium on Geometry Processing (2006)
Konrad Polthier, Alla Sheffer (Editors)

Poisson Surface Reconstruction

Michael Kazhdan[1], Matthew Bolitho[1] and Hugues Hoppe[2]

[1] Johns Hopkins University, Baltimore MD, USA
[2] Microsoft Research, Redmond WA, USA

Abstract

We show that surface reconstruction from oriented points can be cast as a spatial Poisson problem. This Poisson formulation considers all the points at once, without resorting to heuristic spatial partitioning or blending, and is therefore highly resilient to data noise. Unlike radial basis function schemes, our Poisson approach allows a hierarchy of locally supported basis functions, and therefore the solution reduces to a well conditioned sparse linear system. We describe a spatially adaptive multiscale algorithm whose time and space complexities are proportional to the size of the reconstructed model. Experimenting with publicly available scan data, we demonstrate reconstruction of surfaces with greater detail than previously achievable.

1. Introduction

Reconstructing 3D surfaces from point samples is a well studied problem in computer graphics. It allows fitting of scanned data, filling of surface holes, and remeshing of existing models. We provide a novel approach that expresses surface reconstruction as the solution to a Poisson equation.

Like much previous work (Section 2), we approach the problem of surface reconstruction using an implicit function framework. Specifically, like [Kaz05] we compute a 3D *indicator function* χ (defined as 1 at points inside the model, and 0 at points outside), and then obtain the reconstructed surface by extracting an appropriate isosurface.

Our key insight is that there is an integral relationship between oriented points sampled from the surface of a model and the indicator function of the model. Specifically, the gradient of the indicator function is a vector field that is zero almost everywhere (since the indicator function is constant almost everywhere), except at points near the surface, where it is equal to the inward surface normal. Thus, the oriented point samples can be viewed as samples of the gradient of the model's indicator function (Figure 1).

The problem of computing the indicator function thus reduces to inverting the gradient operator, i.e. finding the scalar function χ whose gradient best approximates a vector field $\vec{V}$ defined by the samples, i.e. $\min_\chi \|\nabla\chi - \vec{V}\|$. If we apply the divergence operator, this variational problem transforms into a standard Poisson problem: compute the scalar function χ whose Laplacian (divergence of gradient) equals the divergence of the vector field $\vec{V}$,

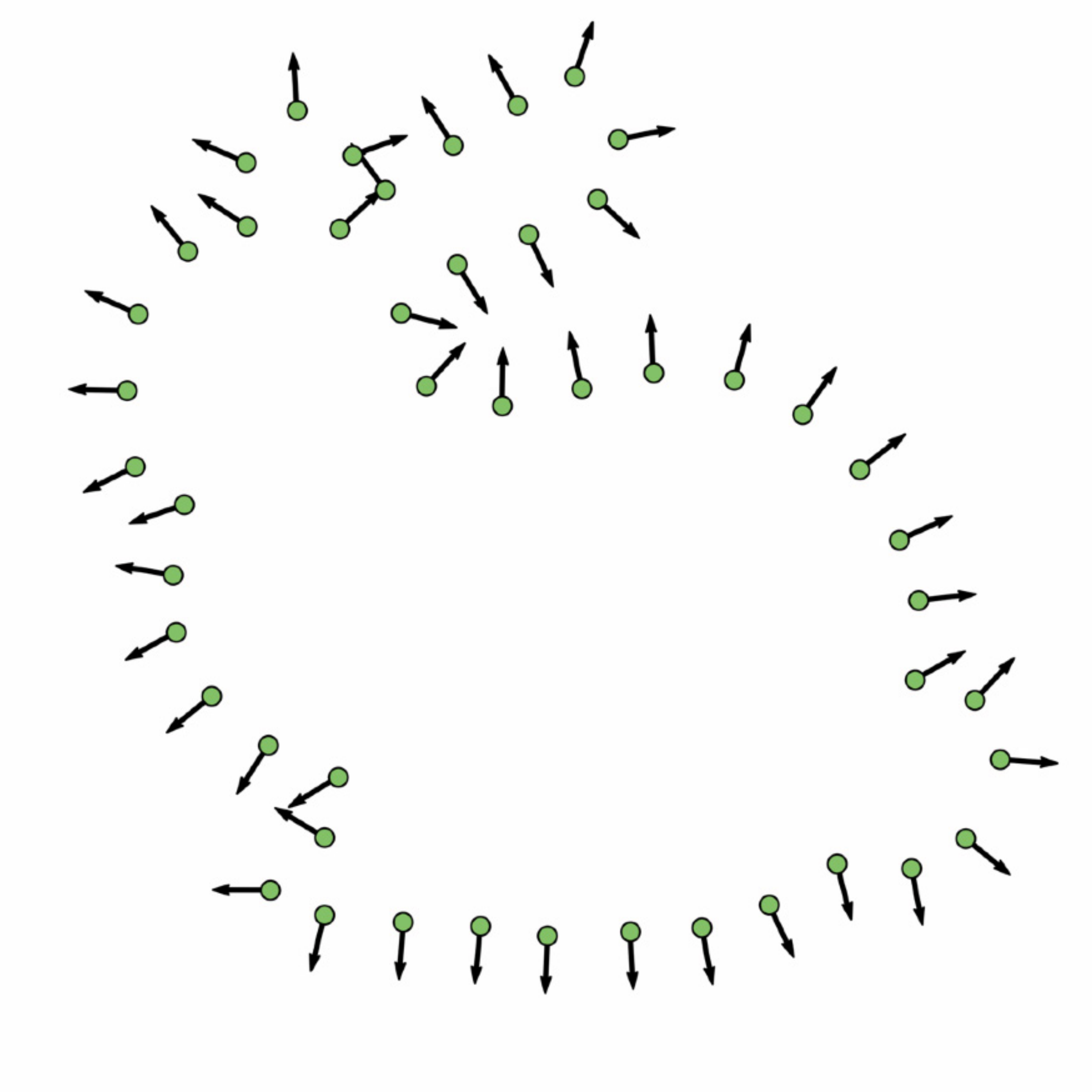
$$\Delta\chi \equiv \nabla\cdot\nabla\chi = \nabla\cdot\vec{V}.$$

We will make these definitions precise in Sections 3 and 4.

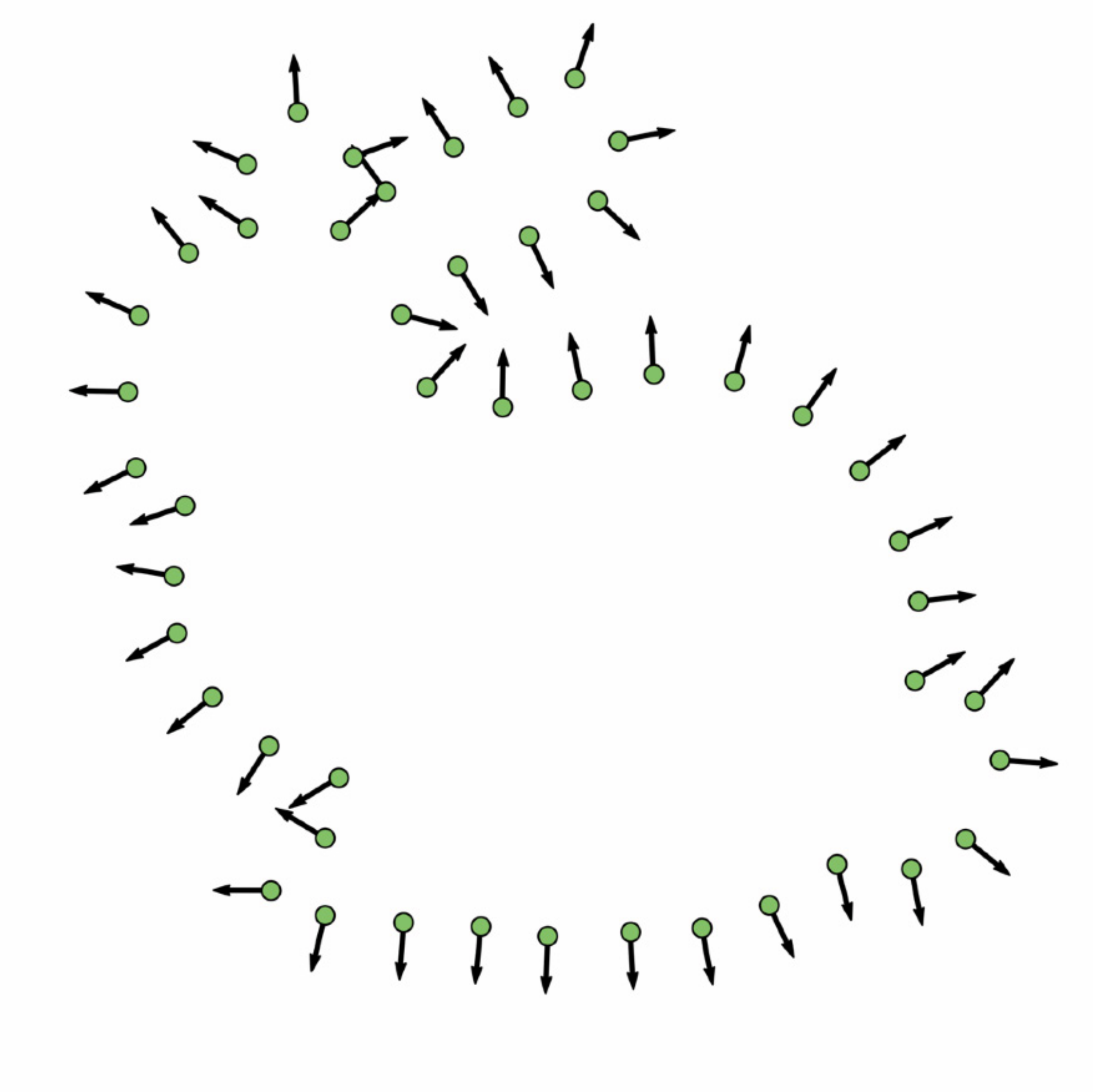
Formulating surface reconstruction as a Poisson problem offers a number of advantages. Many implicit surface fitting methods segment the data into regions for local fitting, and further combine these local approximations using blending functions. In contrast, Poisson reconstruction is a global solution that considers all the data at once, without resorting to heuristic partitioning or blending. Thus, like radial basis function (RBF) approaches, Poisson reconstruction creates very smooth surfaces that robustly approximate noisy data. But, whereas ideal RBFs are globally supported and non-decaying, the Poisson problem admits a hierarchy of *locally supported* functions, and therefore its solution reduces to a well-conditioned sparse linear system.

Figure 1: *Intuitive illustration of Poisson reconstruction in 2D.*

Poisson surface reconstruction

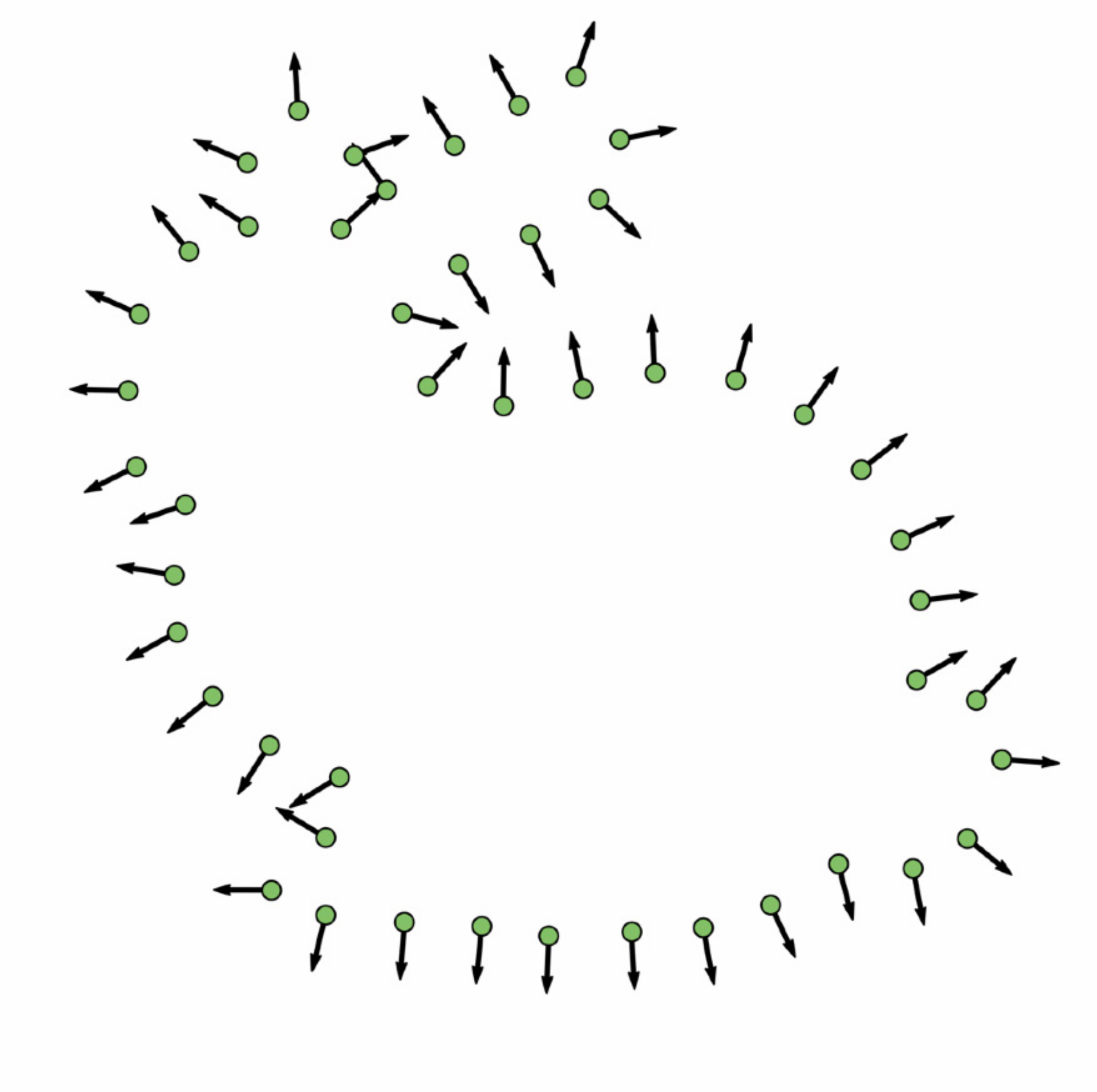


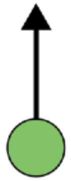
Poisson surface reconstruction



$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

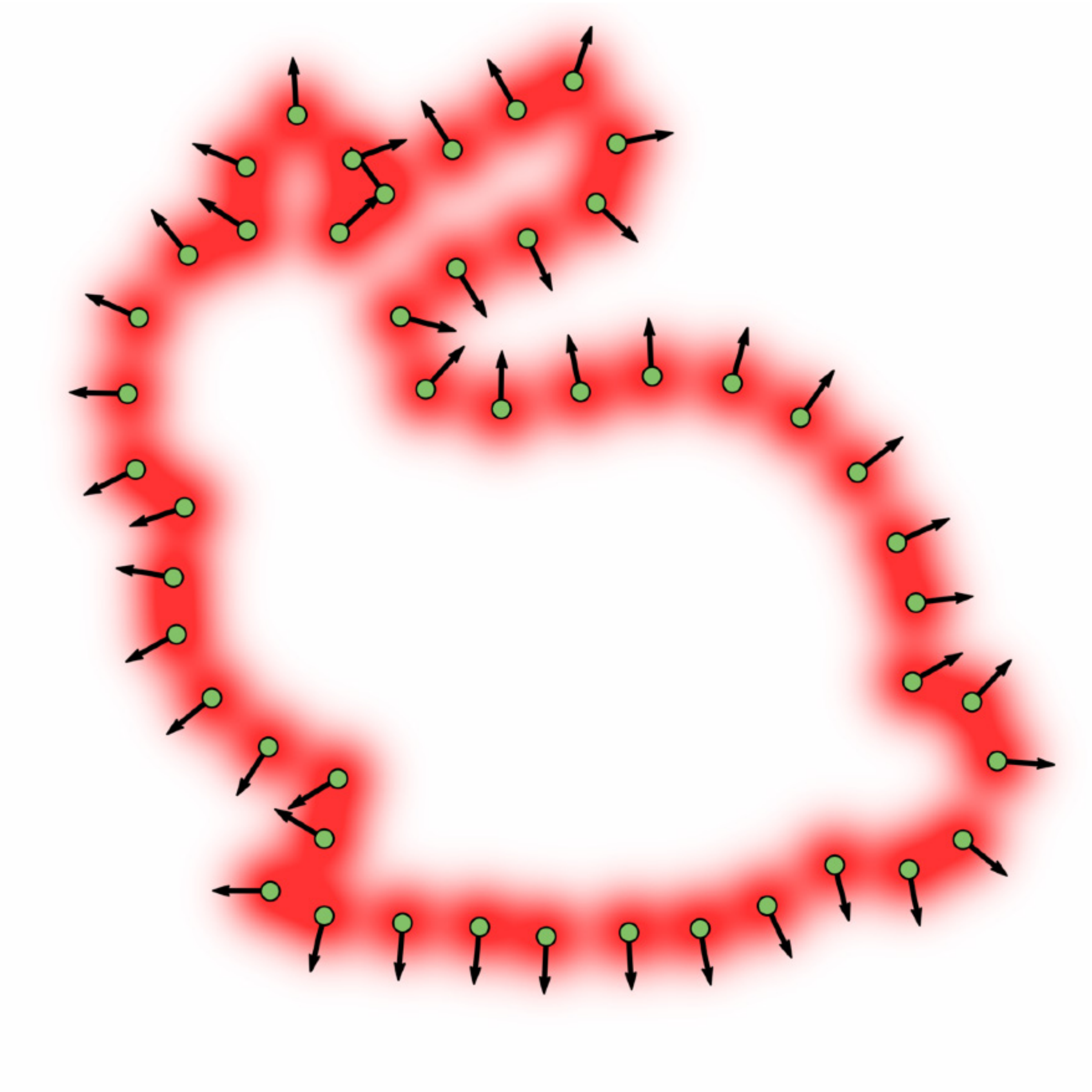
Poisson surface reconstruction

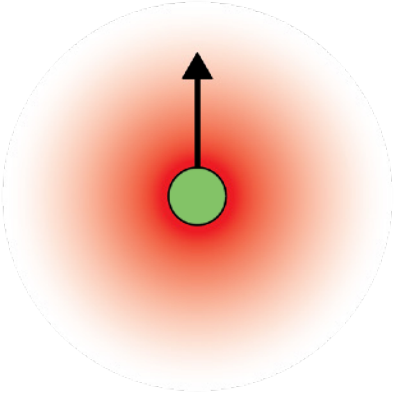




$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

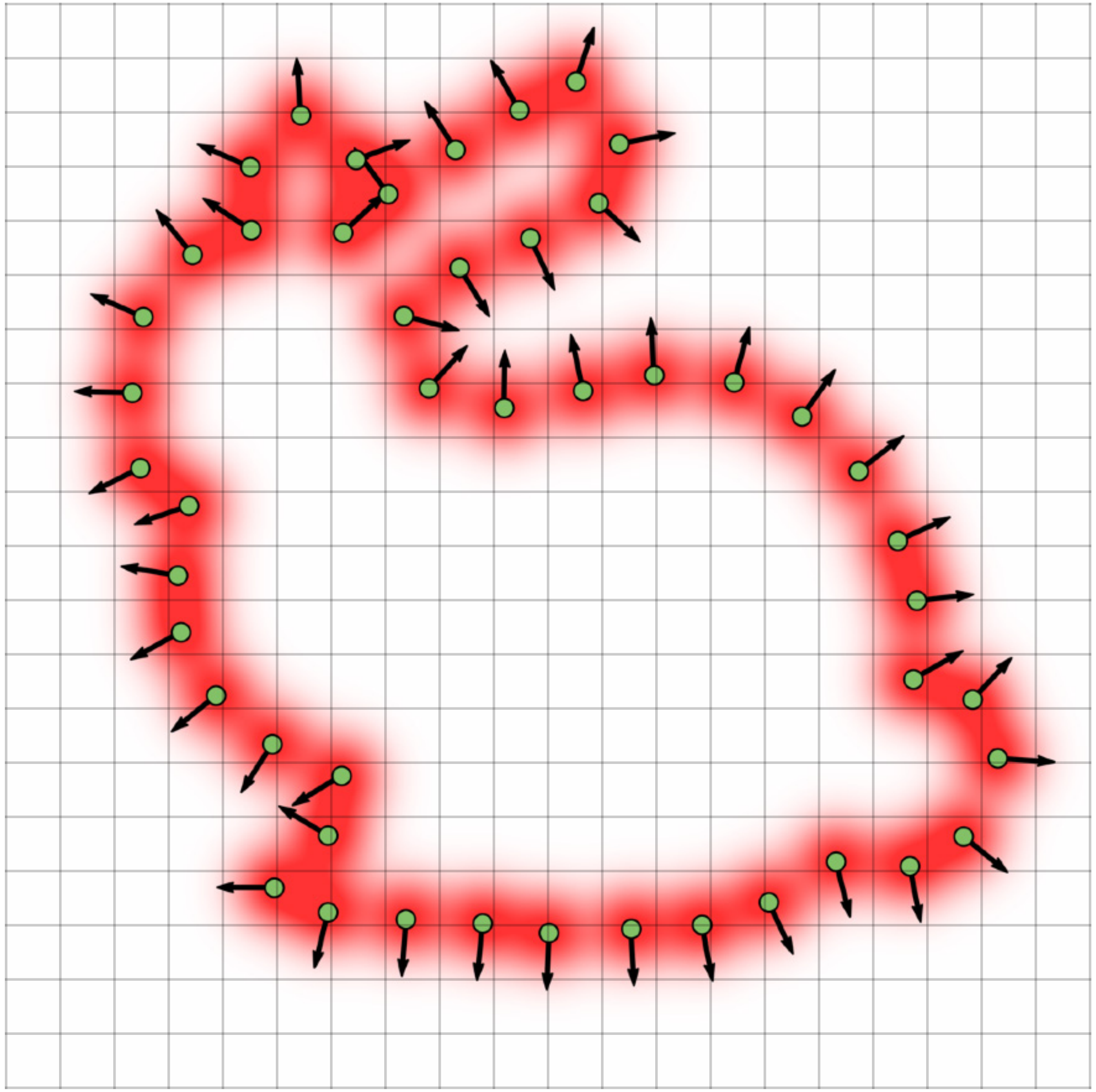
Poisson surface reconstruction

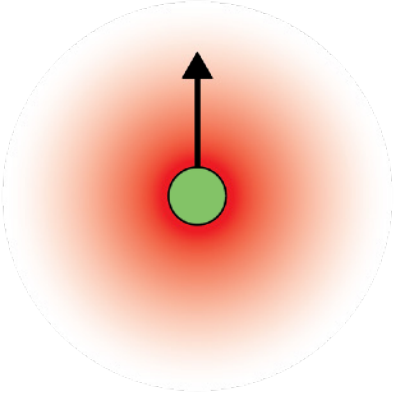




$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

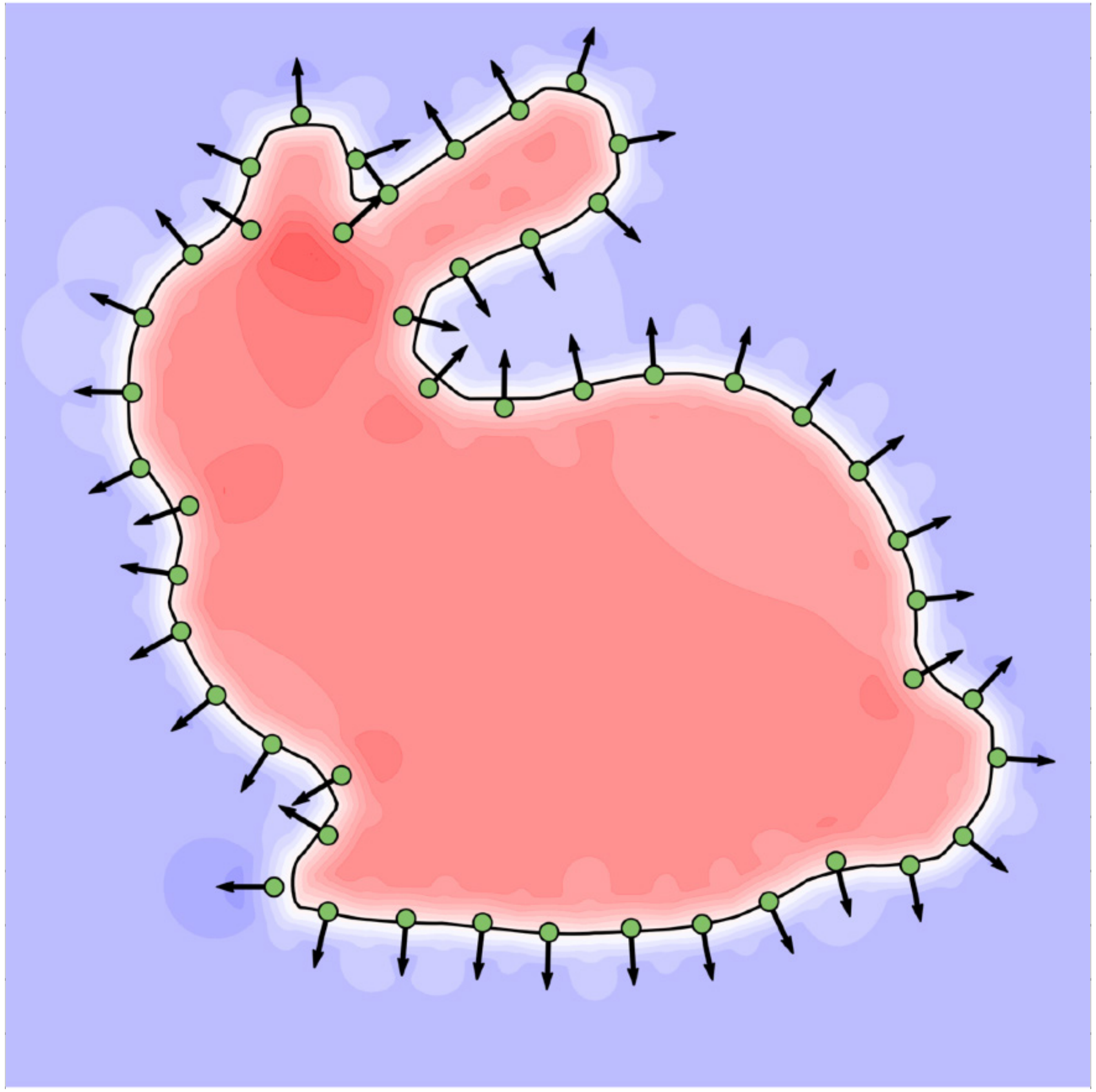
Poisson surface reconstruction

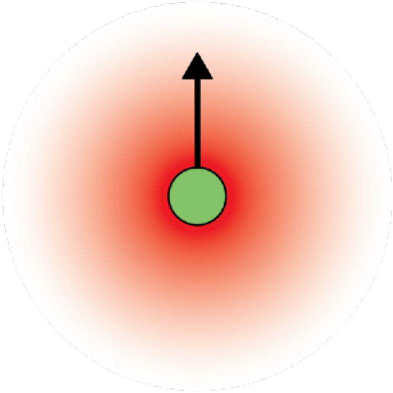




$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

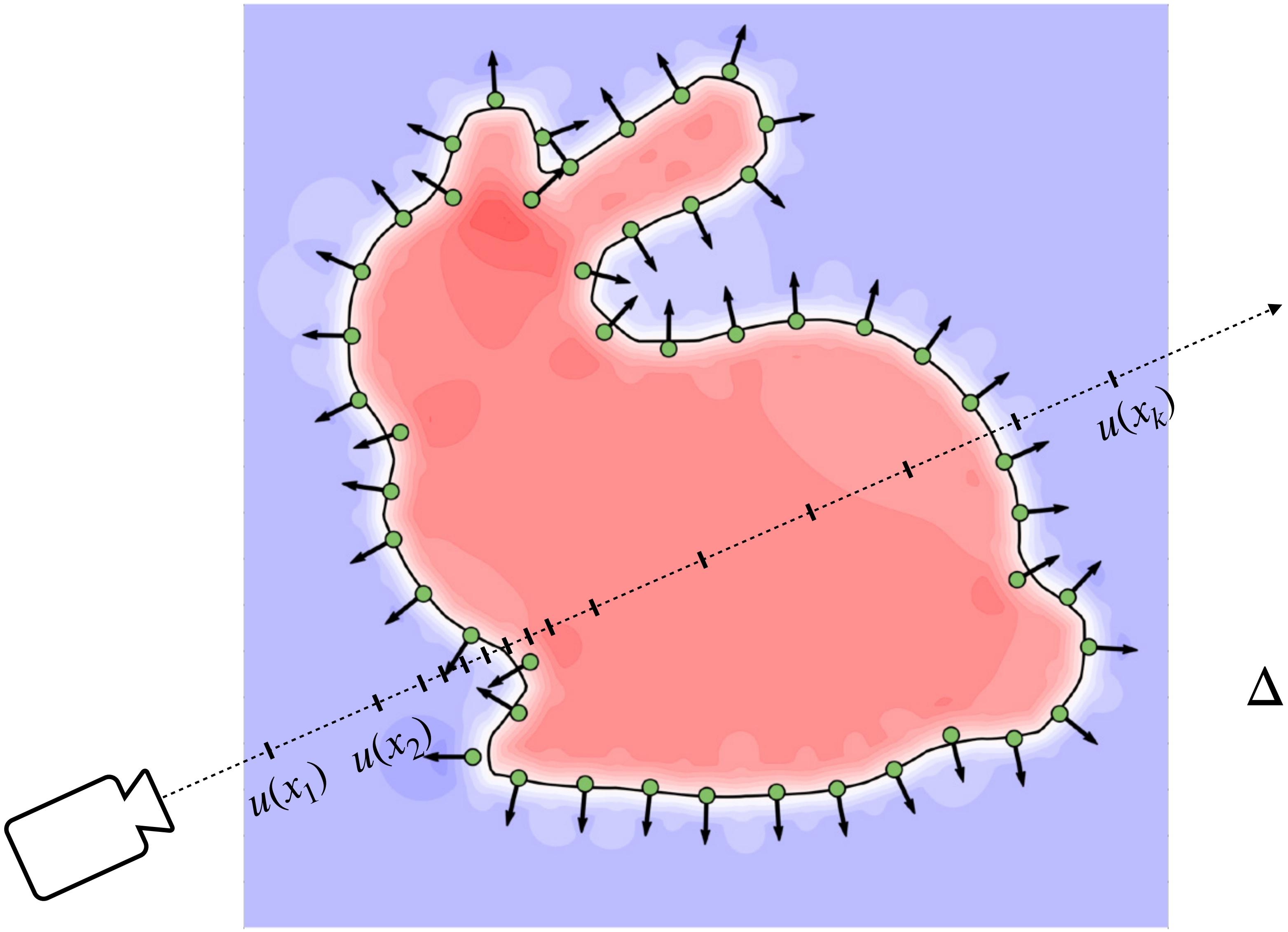
Poisson surface reconstruction

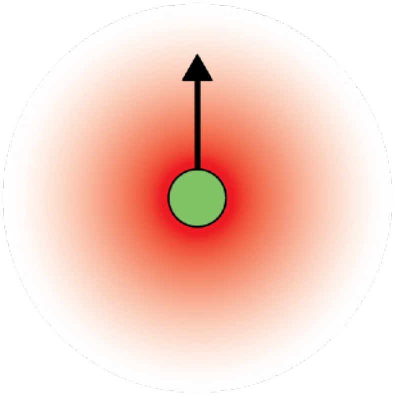




$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

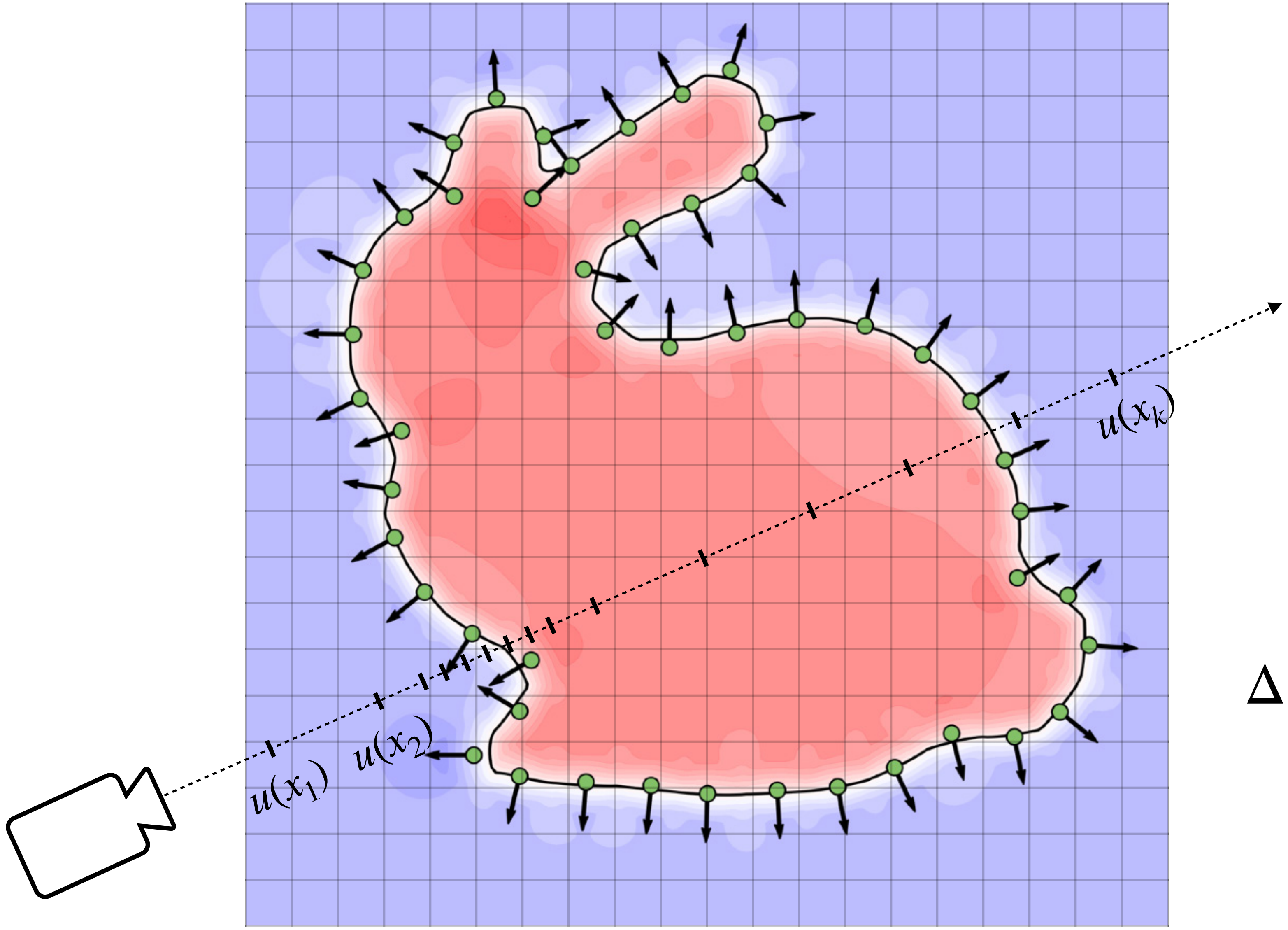
Poisson surface reconstruction

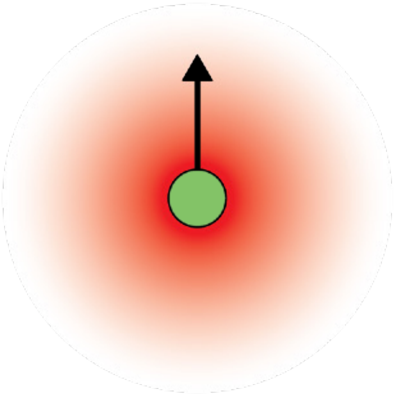




$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

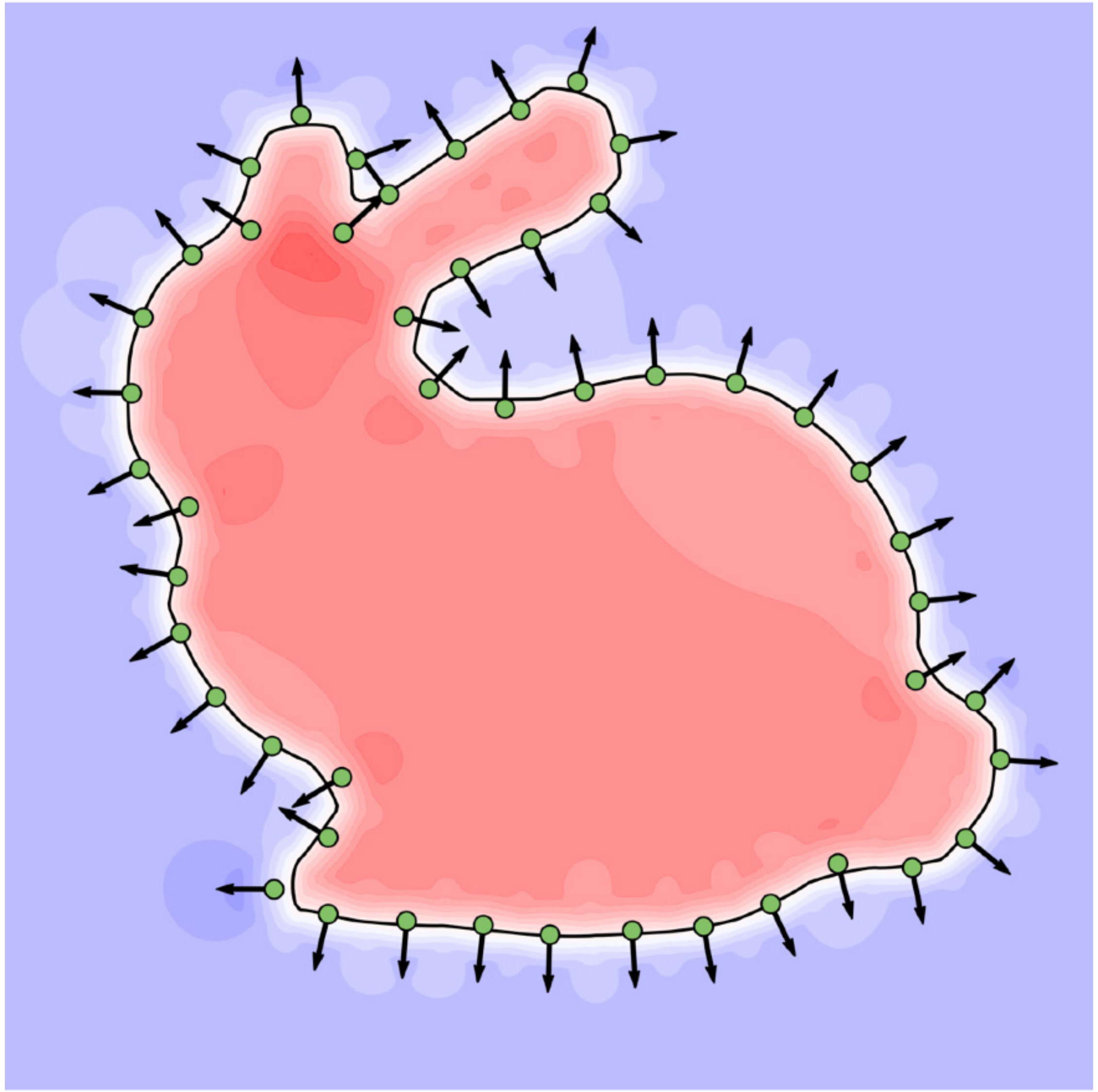
Poisson surface reconstruction

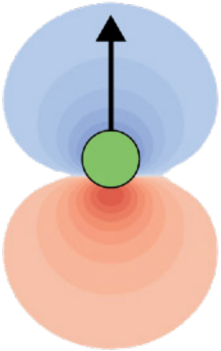




$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

Poisson surface reconstruction



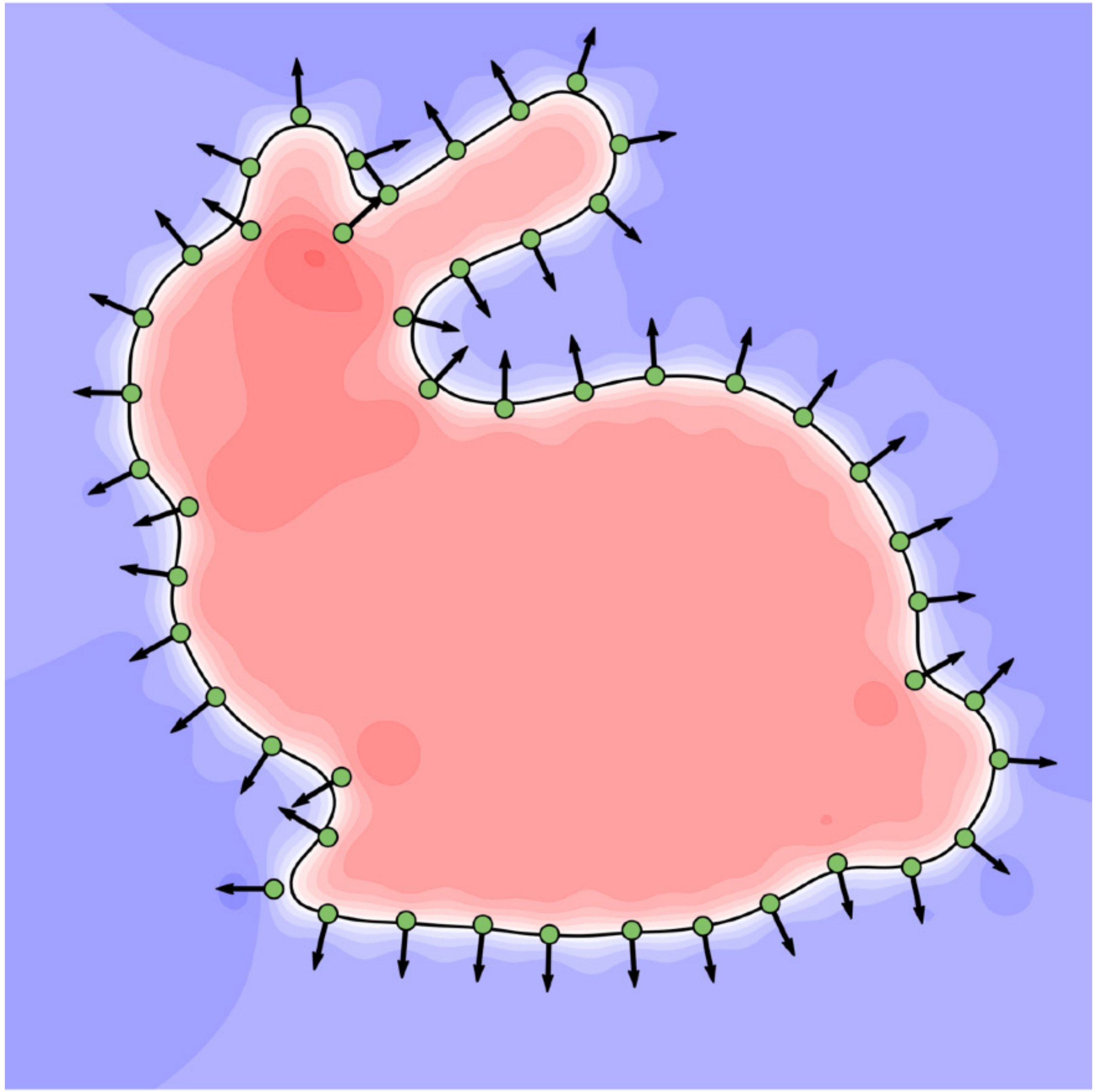


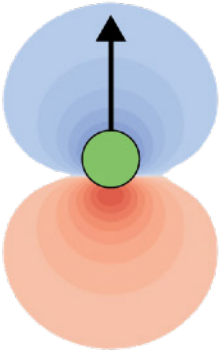
$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

$$\Updownarrow$$

$$u(x) = \sum_i P_\varepsilon(x, y_i, n_i)$$

PSR as a kernel sum



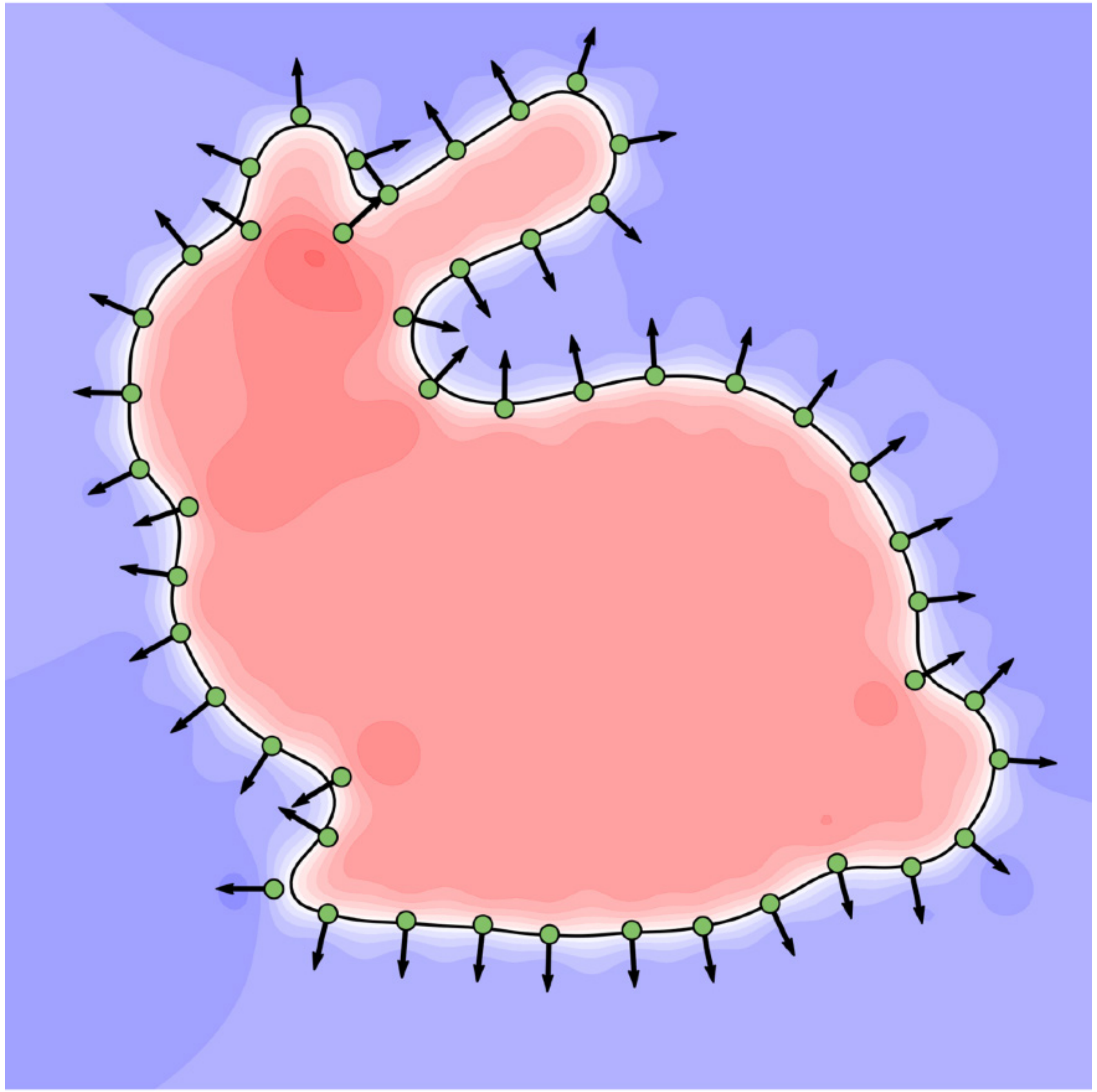


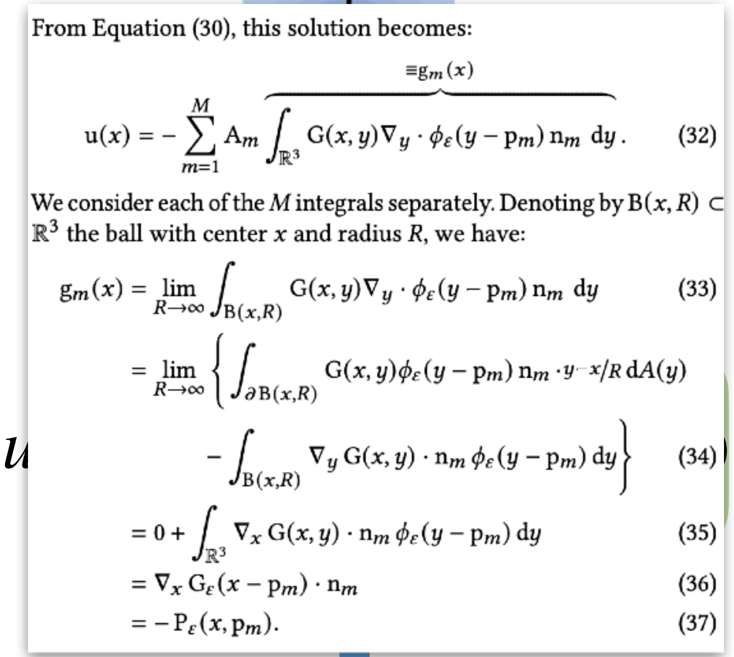
$$\Delta u(x) = \nabla \cdot \left(\sum_i n_i G(x, y_i) \right)$$

$$\Updownarrow$$

$$u(x) = \sum_i P_\varepsilon(x, y_i, n_i)$$

PSR as a kernel sum





From Equation (30), this solution becomes:

$$u(x) = -\sum_{m=1}^{M} A_m \overbrace{\int_{\mathbb{R}^3} G(x,y)\nabla_y \cdot \phi_\varepsilon(y-p_m)\, n_m \, dy}^{\equiv g_m(x)} . \tag{32}$$

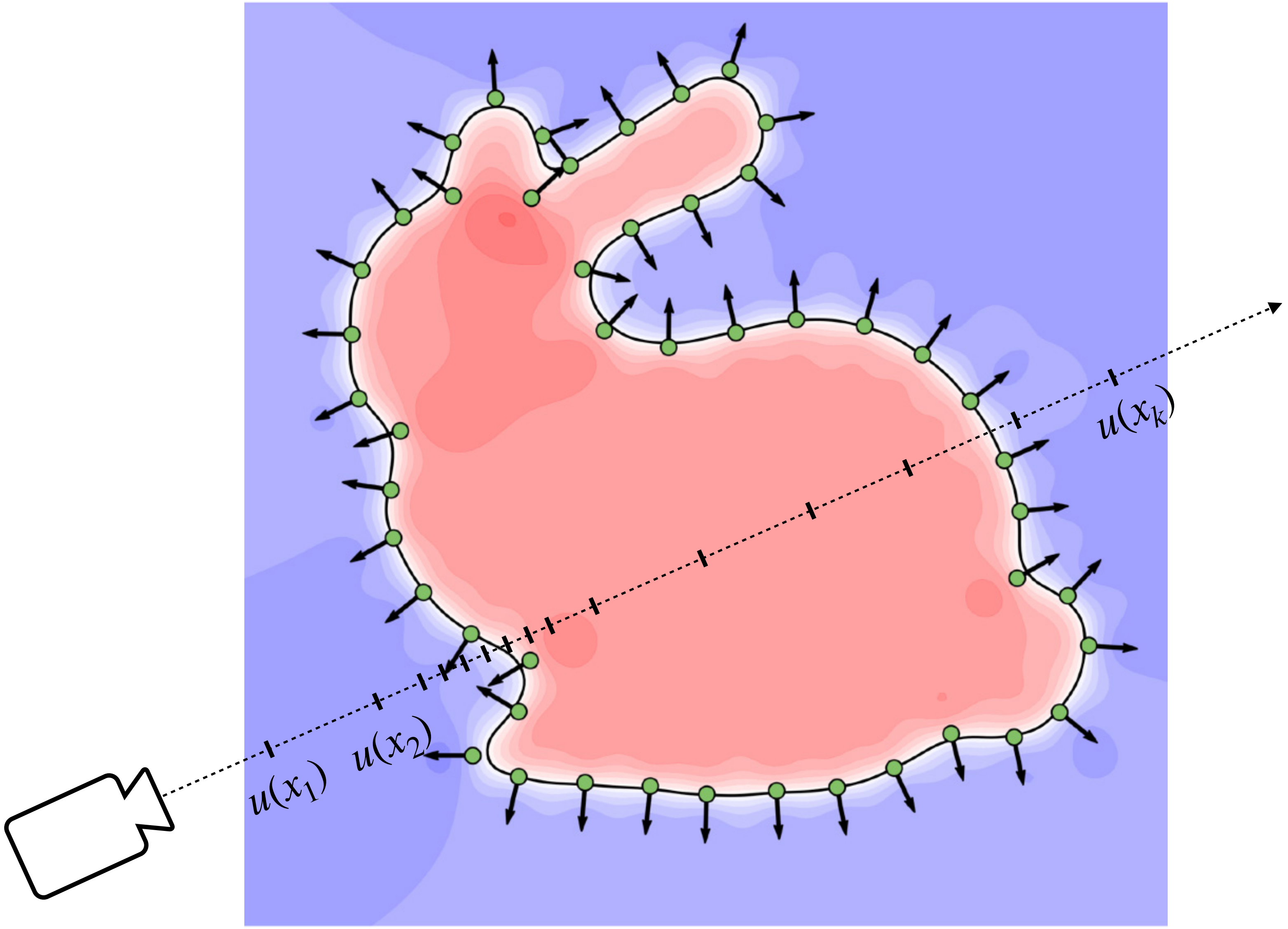
We consider each of the M integrals separately. Denoting by $B(x,R) \subset \mathbb{R}^3$ the ball with center x and radius R, we have:

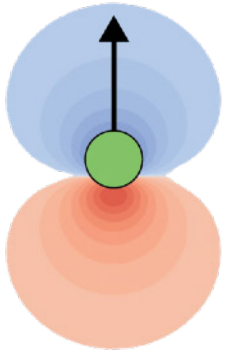
$$\begin{aligned}
g_m(x) &= \lim_{R\to\infty} \int_{B(x,R)} G(x,y)\nabla_y \cdot \phi_\varepsilon(y-p_m)\, n_m \, dy & (33)\\
&= \lim_{R\to\infty} \left\{ \int_{\partial B(x,R)} G(x,y)\phi_\varepsilon(y-p_m)\, n_m \cdot y{-}x/R \, dA(y) \right. \\
&\qquad \left. - \int_{B(x,R)} \nabla_y G(x,y) \cdot n_m \, \phi_\varepsilon(y-p_m)\, dy \right\} & (34)\\
&= 0 + \int_{\mathbb{R}^3} \nabla_x G(x,y) \cdot n_m \, \phi_\varepsilon(y-p_m)\, dy & (35)\\
&= \nabla_x G_\varepsilon(x-p_m) \cdot n_m & (36)\\
&= -P_\varepsilon(x,p_m). & (37)
\end{aligned}$$

Δu

$$u(x) = \sum_i P_\varepsilon(x, y_i, n_i)$$

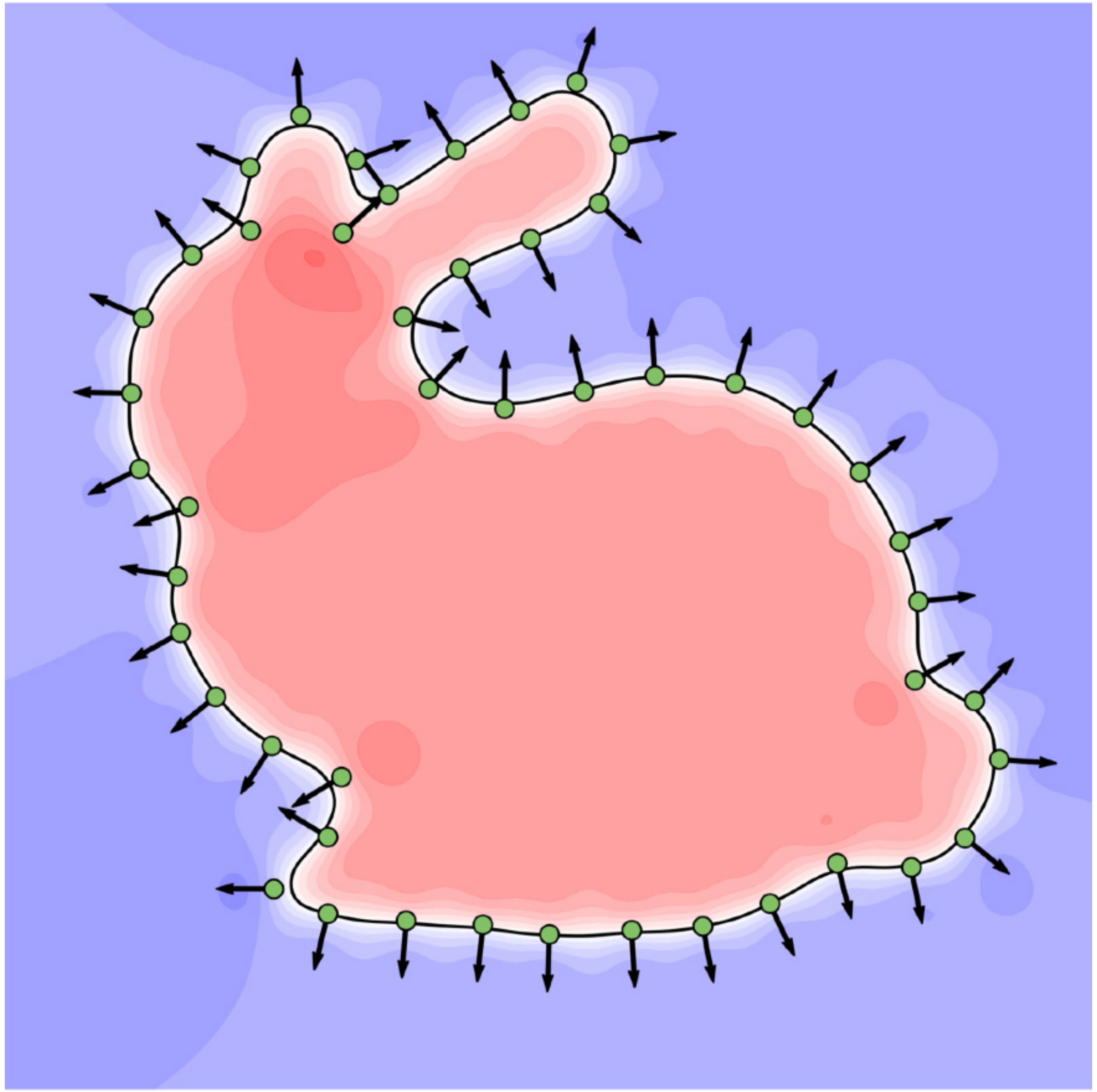
PSR as a kernel sum

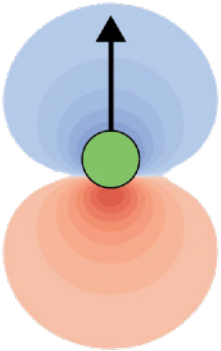




$$u(x) = \sum_i P_\varepsilon(x, y_i, n_i)$$

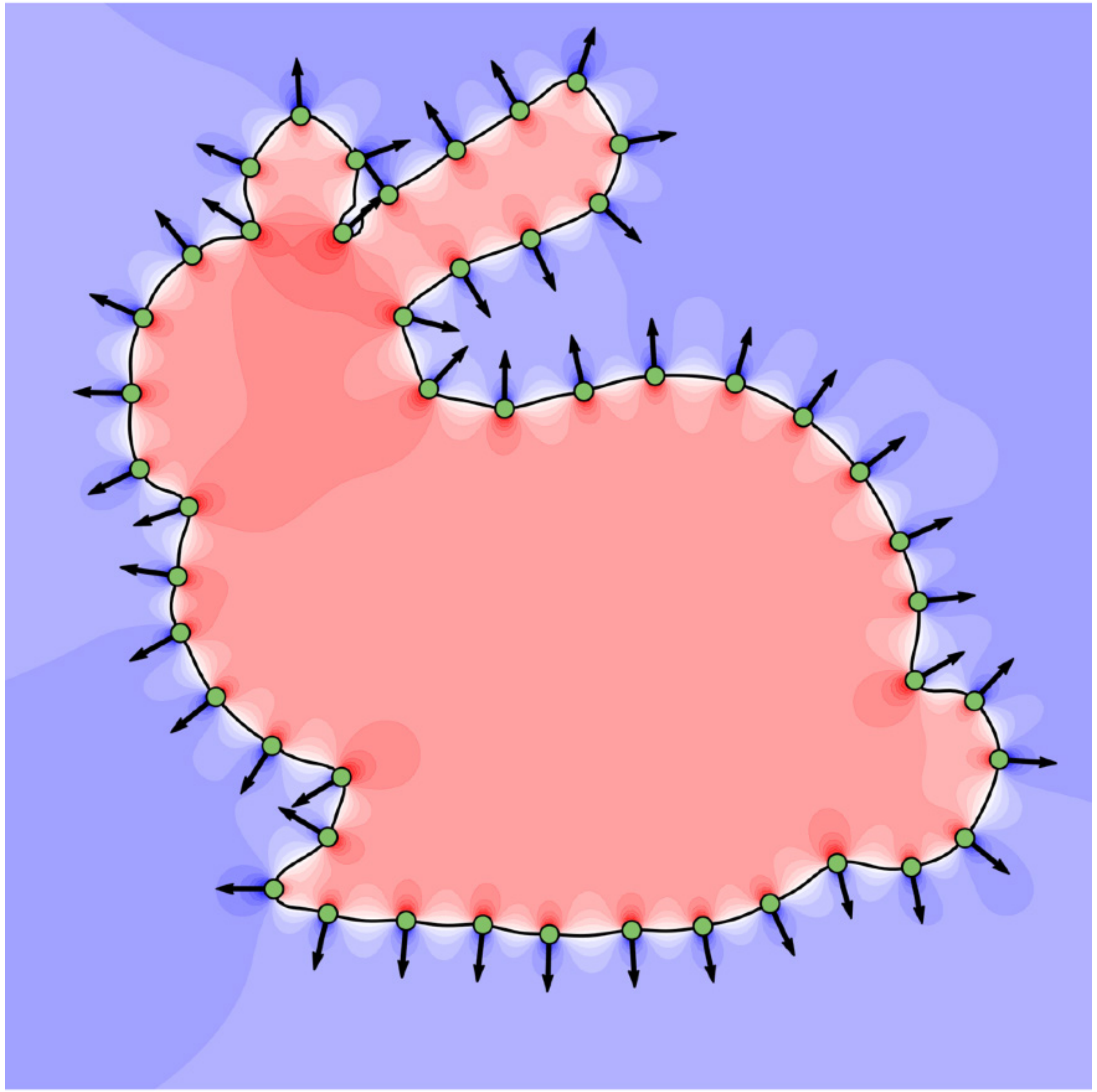
PSR as a kernel sum

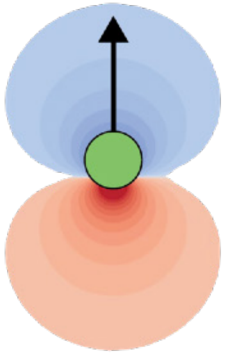




$$u(x) = \sum_i P_{\varepsilon}(x, y_i, n_i)$$

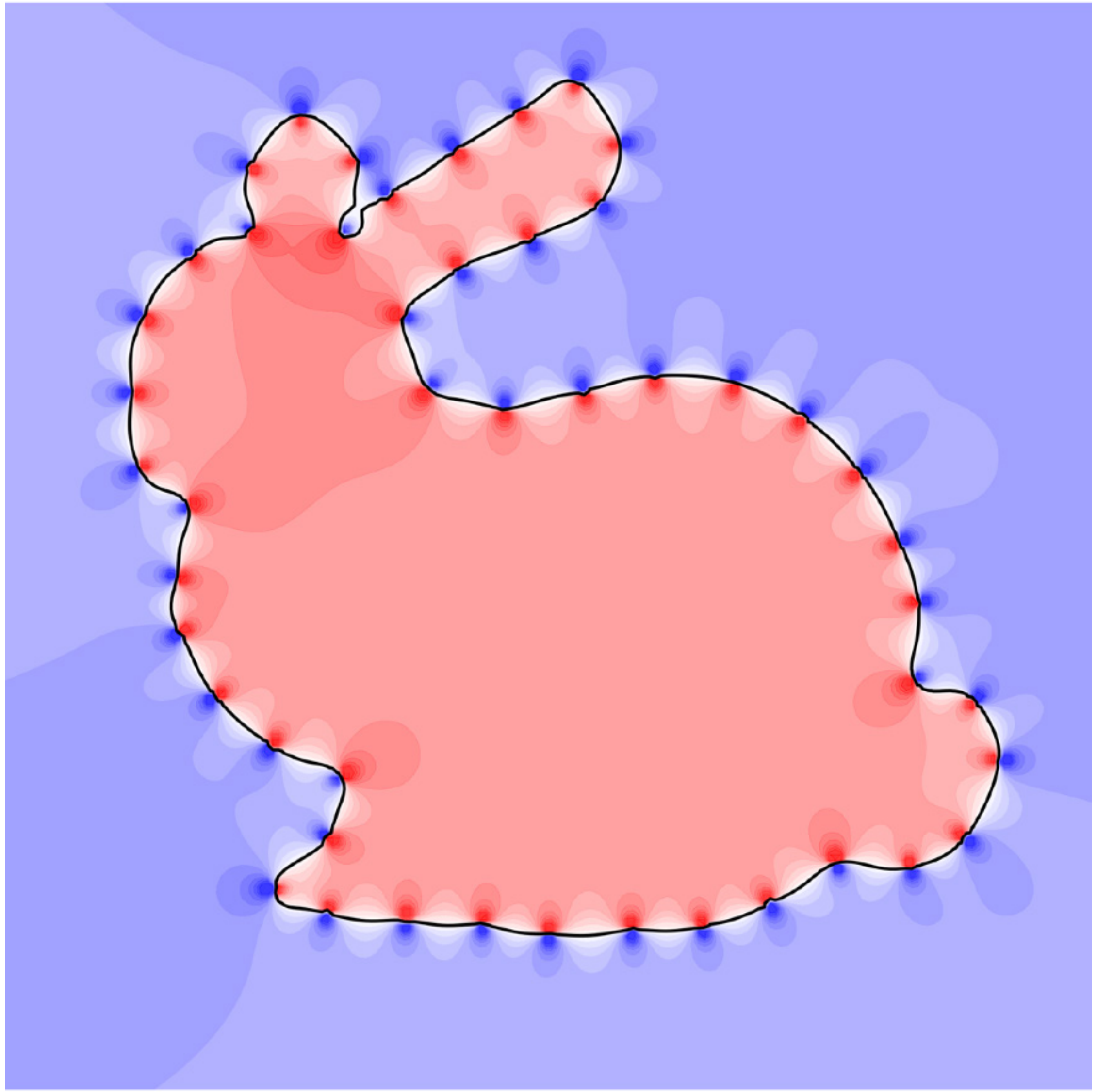
point-cloud winding number

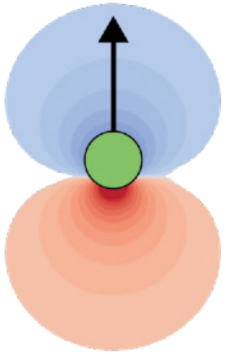




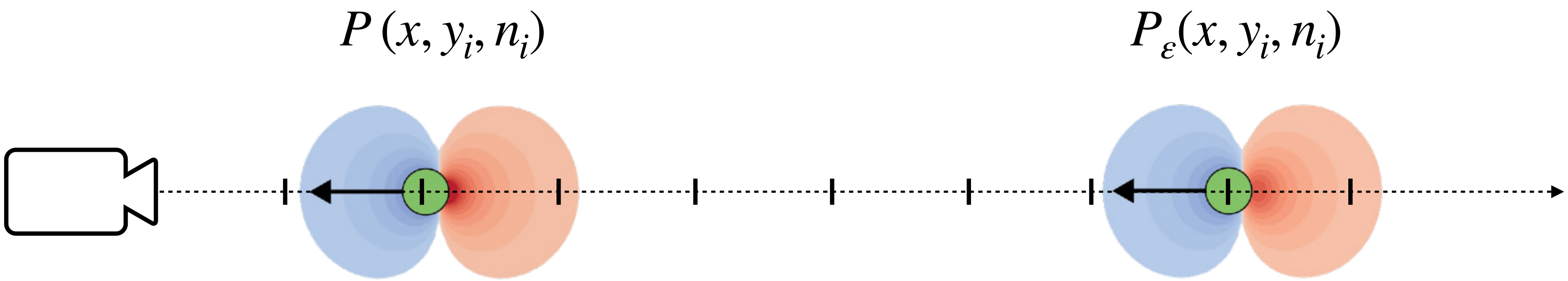
$$u(x) = \sum_i P(x, y_i, n_i)$$

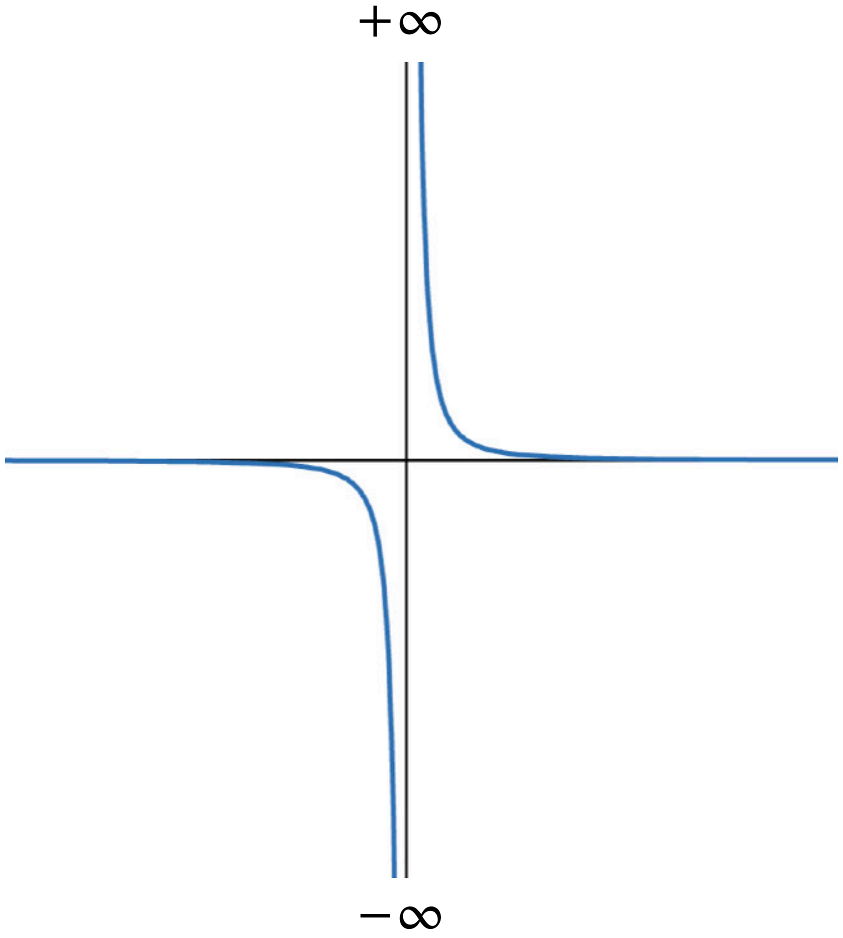
point-cloud winding number



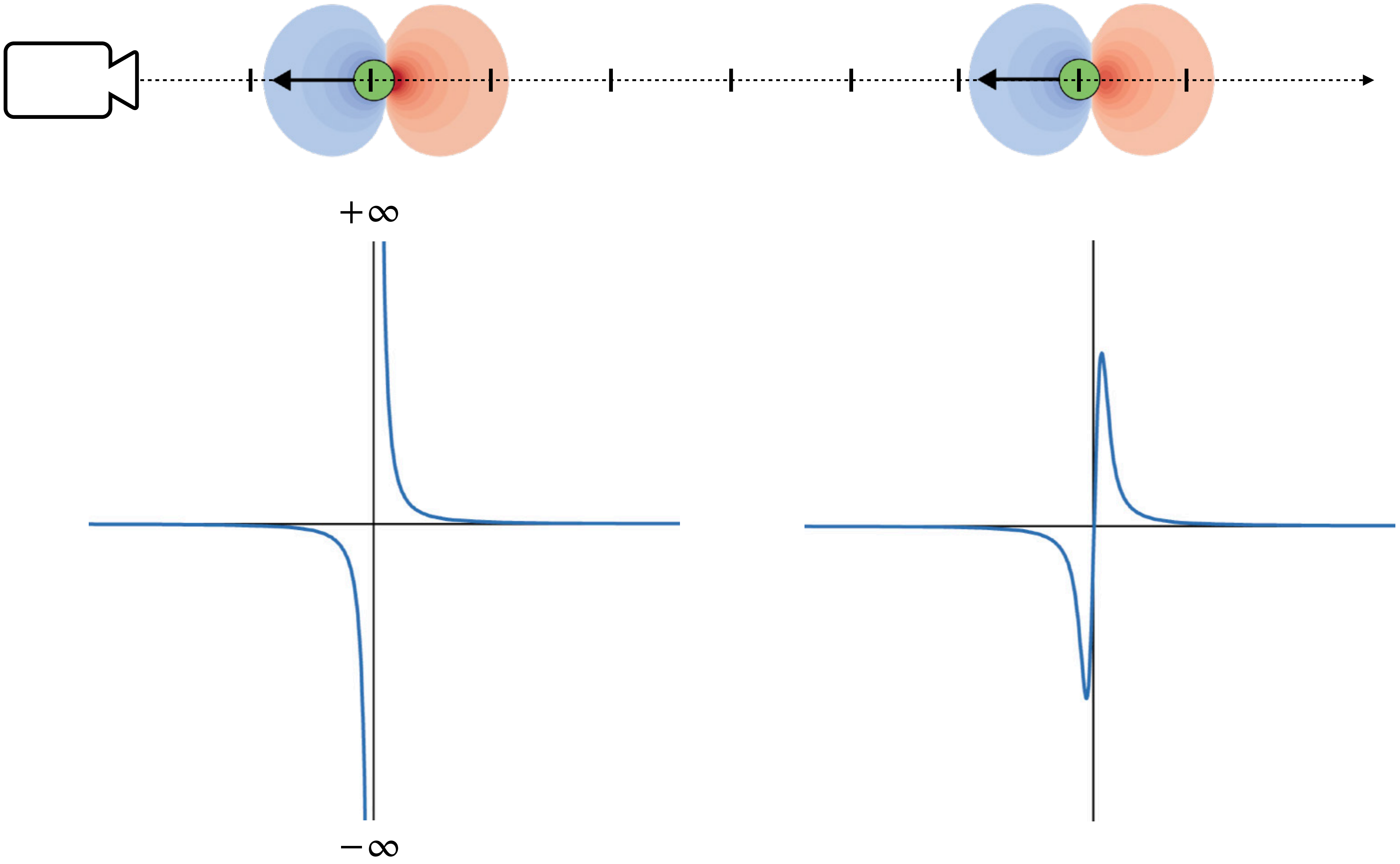


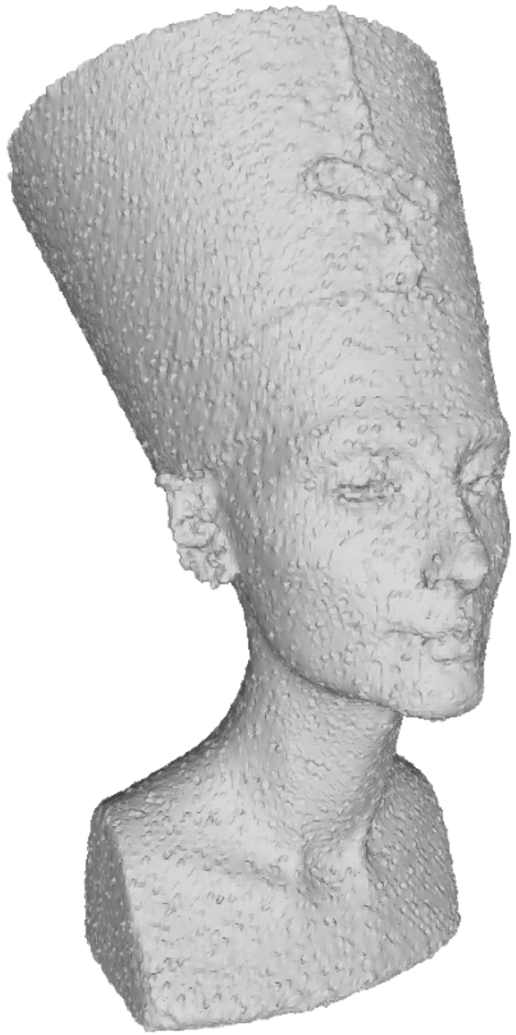
$$u(x) = \sum_i P(x, y_i, n_i)$$


$P\,(x, y_i, n_i)$
$P_\varepsilon(x, y_i, n_i)$

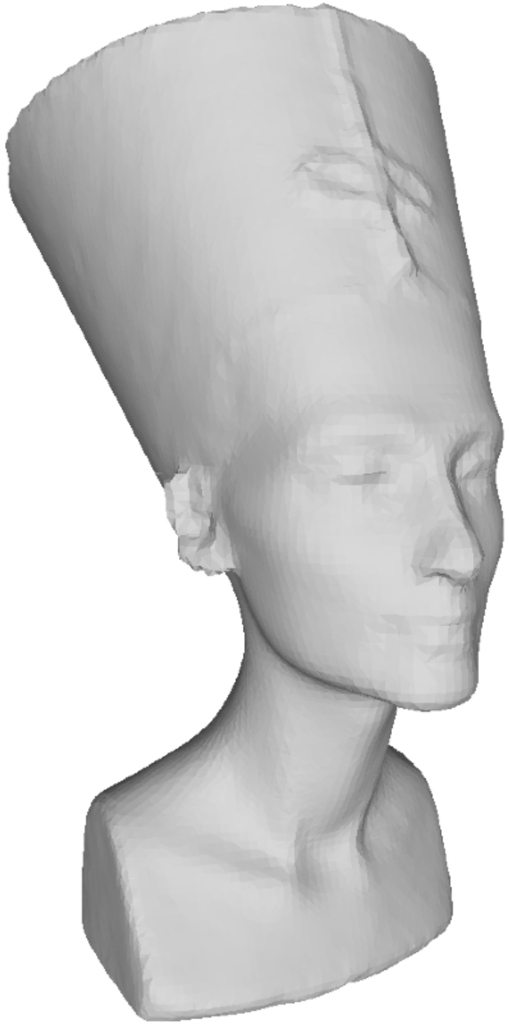

$+\infty$
$-\infty$

$$P\,(x, y_i, n_i) \; * \, G(x, y_i) \quad = \quad P_\varepsilon(x, y_i, n_i)$$



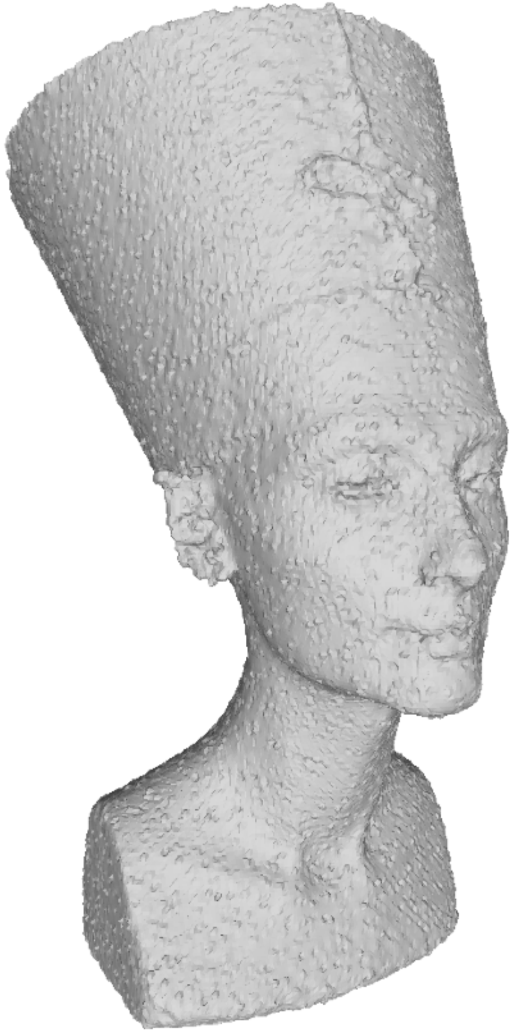


winding number



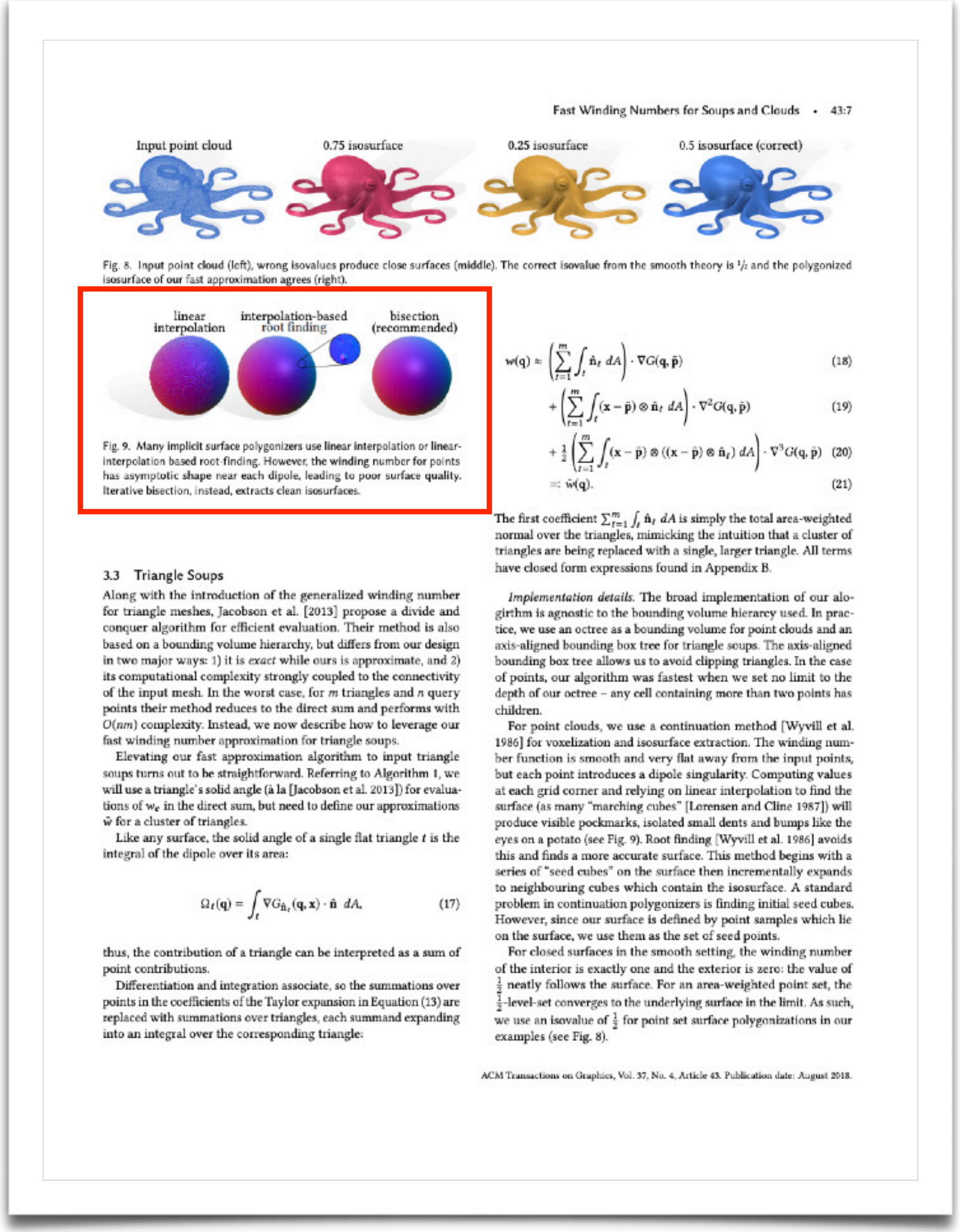
regularized winding number

Barill et al., “Fast Winding Numbers for Soups and Clouds”, SIGGRAPH 2018



winding number

Barill et al., “Fast Winding Numbers for Soups and Clouds”,
SIGGRAPH 2018



Fast Winding Numbers for Soups and Clouds • 43:7

Fig. 8. Input point cloud (left), wrong isovalues produce close surfaces (middle). The correct isovalue from the smooth theory is ½ and the polygonized isosurface of our fast approximation agrees (right).

Fig. 9. Many implicit surface polygonizers use linear interpolation or linear-interpolation based root-finding. However, the winding number for points has asymptotic shape near each dipole, leading to poor surface quality. Iterative bisection, instead, extracts clean isosurfaces.

3.3 Triangle Soups

Along with the introduction of the generalized winding number for triangle meshes, Jacobson et al. [2013] propose a divide and conquer algorithm for efficient evaluation. Their method is also based on a bounding volume hierarchy, but differs from our design in two major ways: 1) it is *exact* while ours is approximate, and 2) its computational complexity strongly coupled to the connectivity of the input mesh. In the worst case, for m triangles and n query points their method reduces to the direct sum and performs with $O(nm)$ complexity. Instead, we now describe how to leverage our fast winding number approximation for triangle soups.

Elevating our fast approximation algorithm to input triangle soups turns out to be straightforward. Referring to Algorithm 1, we will use a triangle's solid angle (à la [Jacobson et al. 2013]) for evaluations of w_e in the direct sum, but need to define our approximations $\tilde{w}$ for a cluster of triangles.

Like any surface, the solid angle of a single flat triangle t is the integral of the dipole over its area:

$$\Omega_t(\mathbf{q}) = \int_t \nabla G_{\hat{\mathbf{n}}_t}(\mathbf{q}, \mathbf{x}) \cdot \hat{\mathbf{n}} \;\; dA, \tag{17}$$

thus, the contribution of a triangle can be interpreted as a sum of point contributions.

Differentiation and integration associate, so the summations over points in the coefficients of the Taylor expansion in Equation (13) are replaced with summations over triangles, each summand expanding into an integral over the corresponding triangle:

$$w(\mathbf{q}) \approx \left(\sum_{t=1}^{m} \int_t \hat{\mathbf{n}}_t \; dA\right) \cdot \nabla G(\mathbf{q}, \tilde{\mathbf{p}}) \tag{18}$$

$$+ \left(\sum_{t=1}^{m} \int_t (\mathbf{x} - \tilde{\mathbf{p}}) \otimes \hat{\mathbf{n}}_t \; dA\right) \cdot \nabla^2 G(\mathbf{q}, \tilde{\mathbf{p}}) \tag{19}$$

$$+ \tfrac{1}{2}\left(\sum_{t=1}^{m} \int_t (\mathbf{x} - \tilde{\mathbf{p}}) \otimes ((\mathbf{x} - \tilde{\mathbf{p}}) \otimes \hat{\mathbf{n}}_t) \; dA\right) \cdot \nabla^3 G(\mathbf{q}, \tilde{\mathbf{p}}) \tag{20}$$

$$=: \tilde{w}(\mathbf{q}). \tag{21}$$

The first coefficient $\sum_{t=1}^{m} \int_t \hat{\mathbf{n}}_t \; dA$ is simply the total area-weighted normal over the triangles, mimicking the intuition that a cluster of triangles are being replaced with a single, larger triangle. All terms have closed form expressions found in Appendix B.

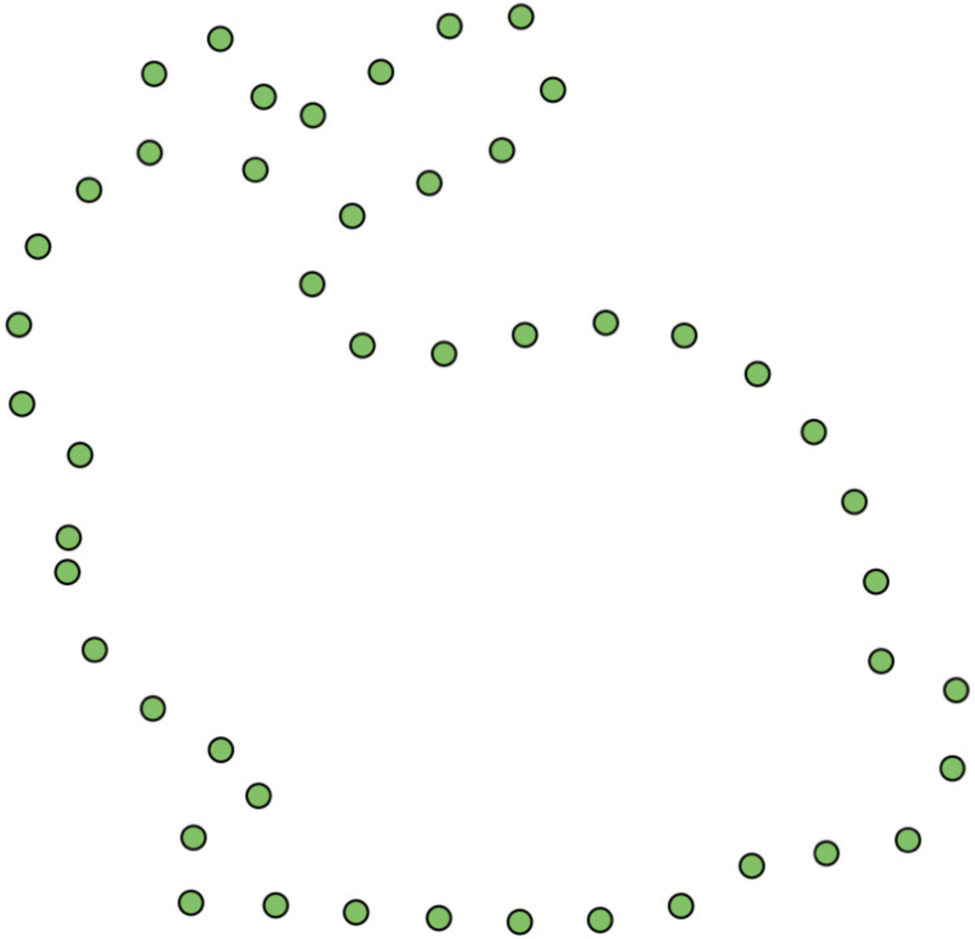
Implementation details. The broad implementation of our alogirthm is agnostic to the bounding volume hierarcy used. In practice, we use an octree as a bounding volume for point clouds and an axis-aligned bounding box tree for triangle soups. The axis-aligned bounding box tree allows us to avoid clipping triangles. In the case of points, our algorithm was fastest when we set no limit to the depth of our octree – any cell containing more than two points has children.

For point clouds, we use a continuation method [Wyvill et al. 1986] for voxelization and isosurface extraction. The winding number function is smooth and very flat away from the input points, but each point introduces a dipole singularity. Computing values at each grid corner and relying on linear interpolation to find the surface (as many "marching cubes" [Lorensen and Cline 1987]) will produce visible pockmarks, isolated small dents and bumps like the eyes on a potato (see Fig. 9). Root finding [Wyvill et al. 1986] avoids this and finds a more accurate surface. This method begins with a series of "seed cubes" on the surface then incrementally expands to neighbouring cubes which contain the isosurface. A standard problem in continuation polygonizers is finding initial seed cubes. However, since our surface is defined by point samples which lie on the surface, we use them as the set of seed points.

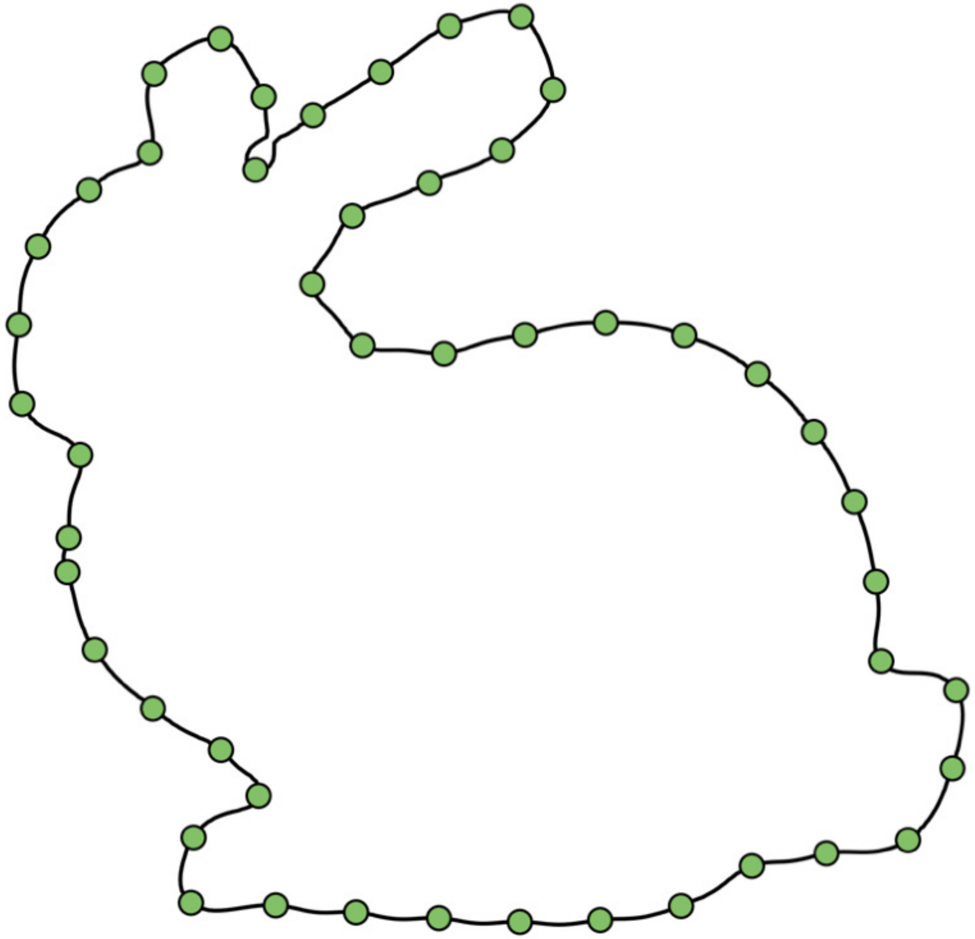
For closed surfaces in the smooth setting, the winding number of the interior is exactly one and the exterior is zero: the value of $\frac{1}{2}$ neatly follows the surface. For an area-weighted point set, the $\frac{1}{2}$-level-set converges to the underlying surface in the limit. As such, we use an isovalue of $\frac{1}{2}$ for point set surface polygonizations in our examples (see Fig. 8).

ACM Transactions on Graphics, Vol. 37, No. 4, Article 43. Publication date: August 2018.

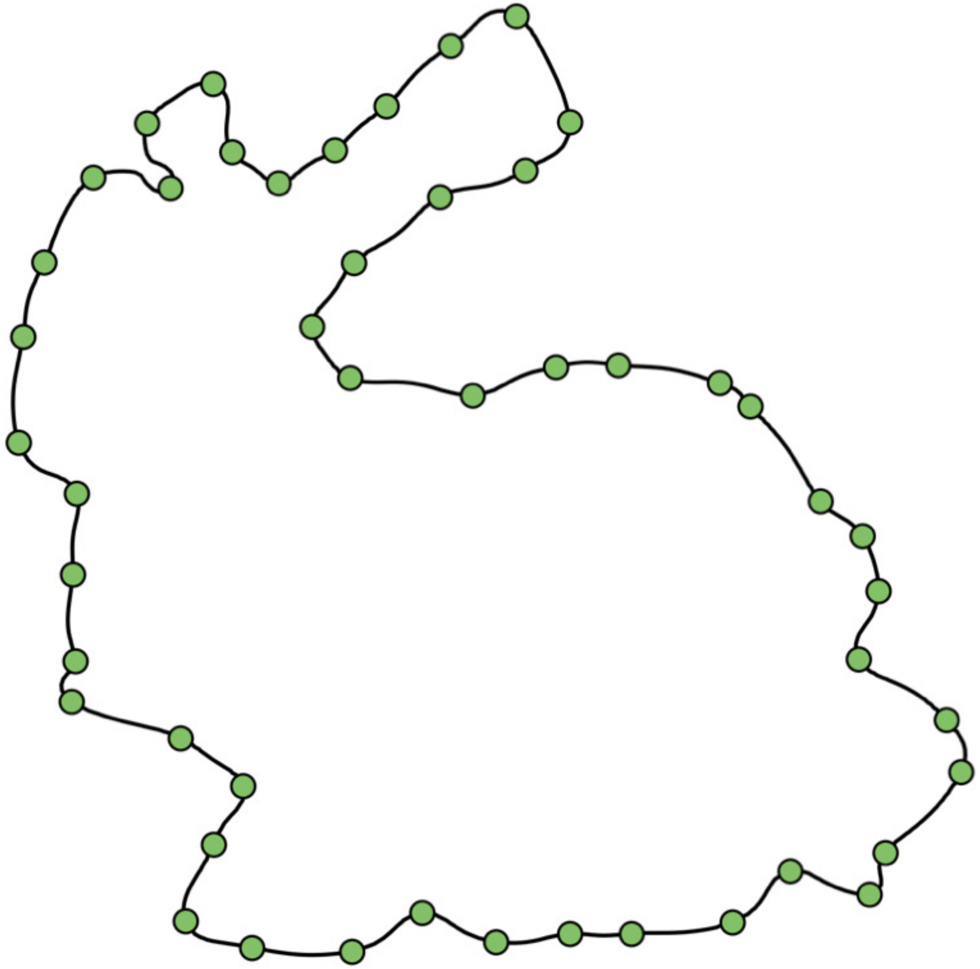
point-cloud winding number



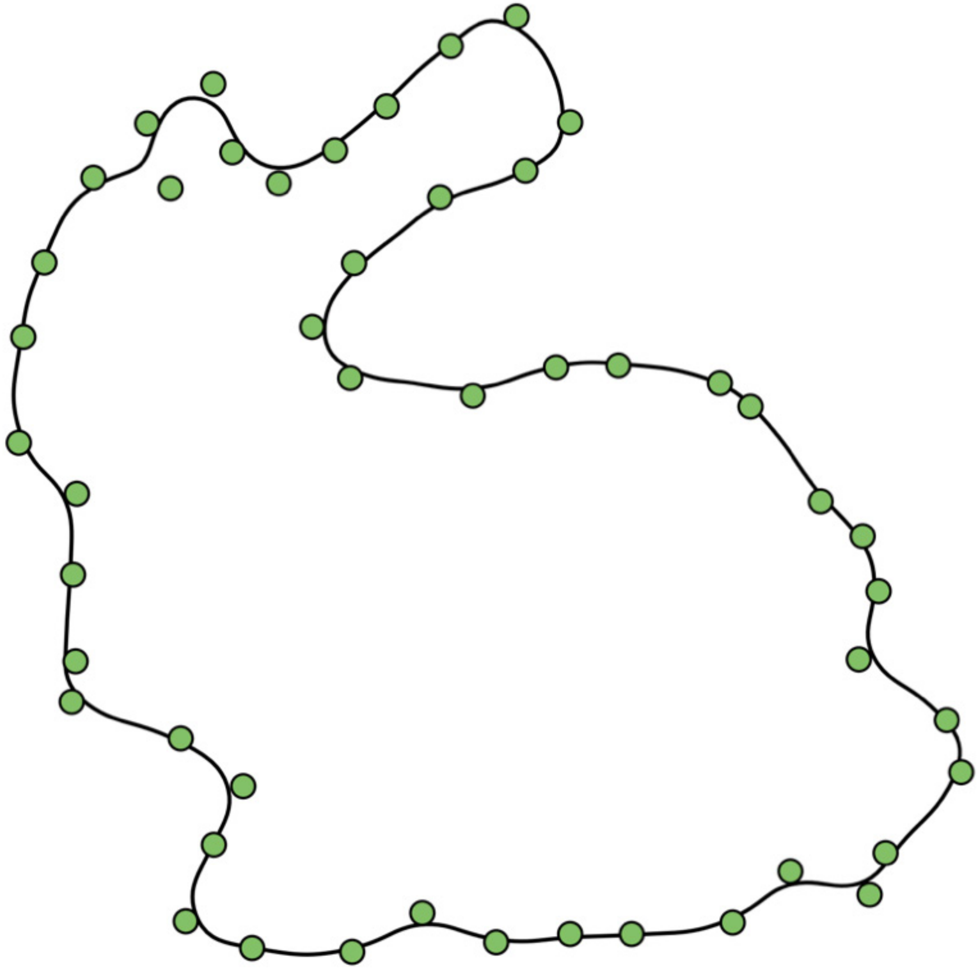
point-cloud winding number



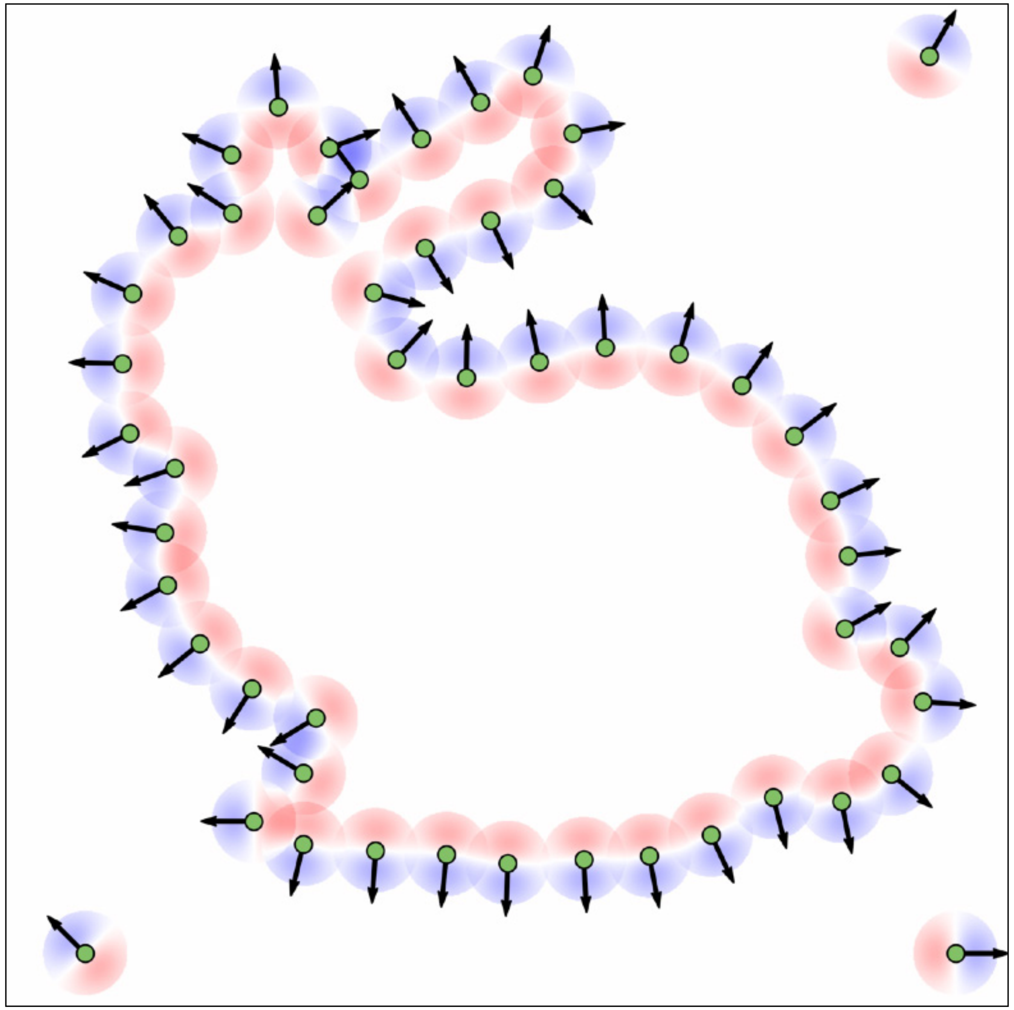
point-cloud winding number



regularized winding number

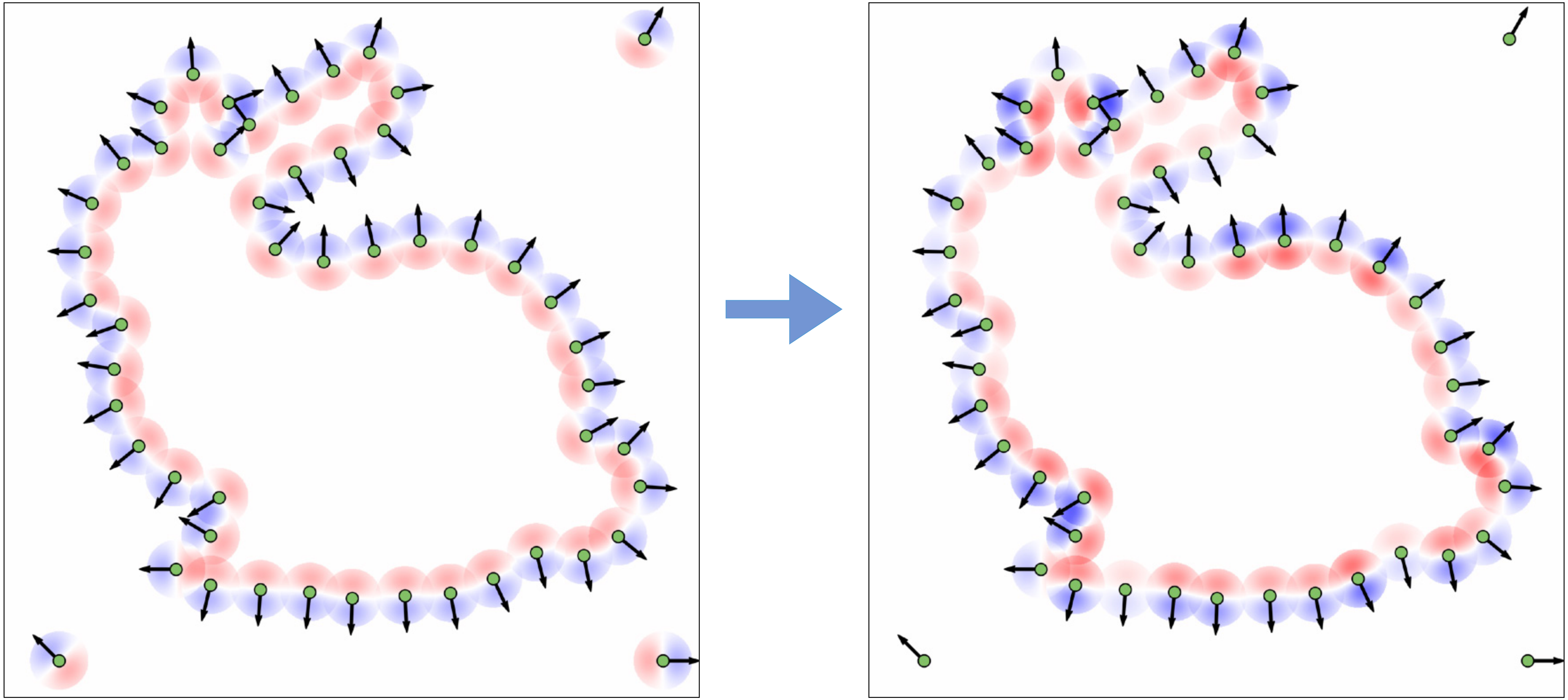


regularized winding number



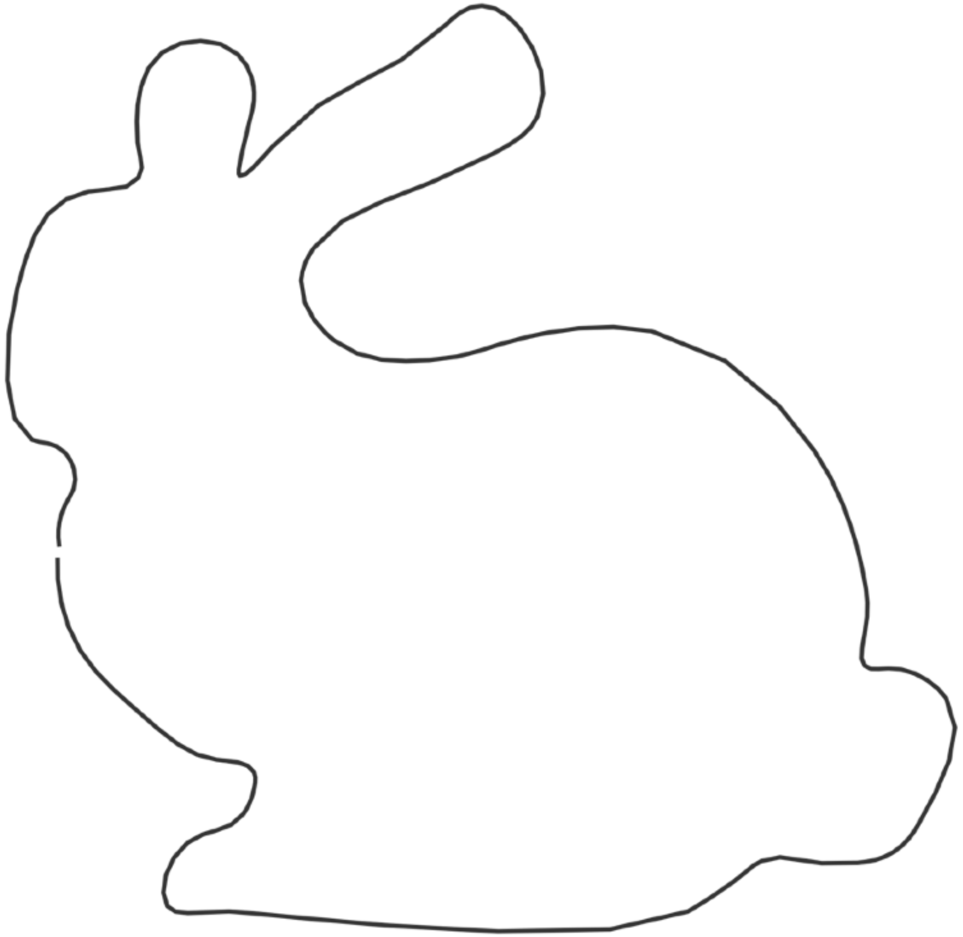
$$\sum P_{\varepsilon}\left(x, y_i, n_i\right)$$

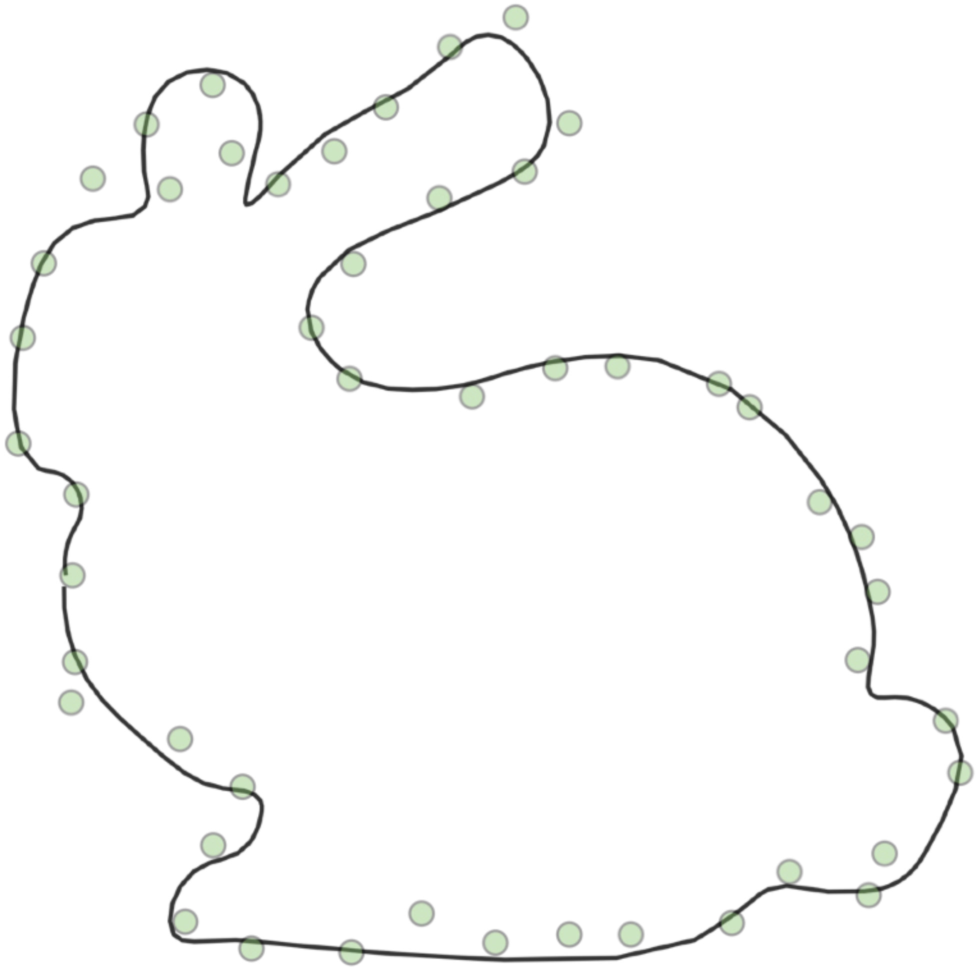
regularized *dipole sum*

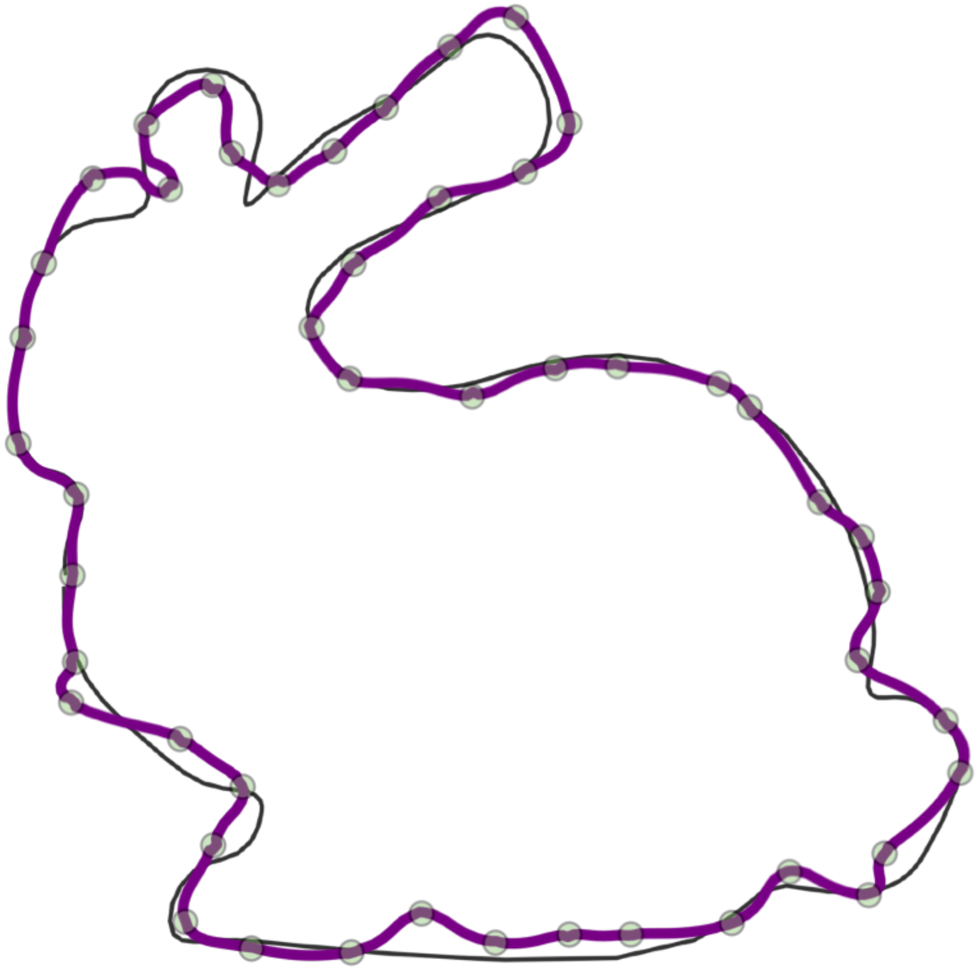


$$\sum P_{\varepsilon}\left(x, y_i, n_i\right)$$

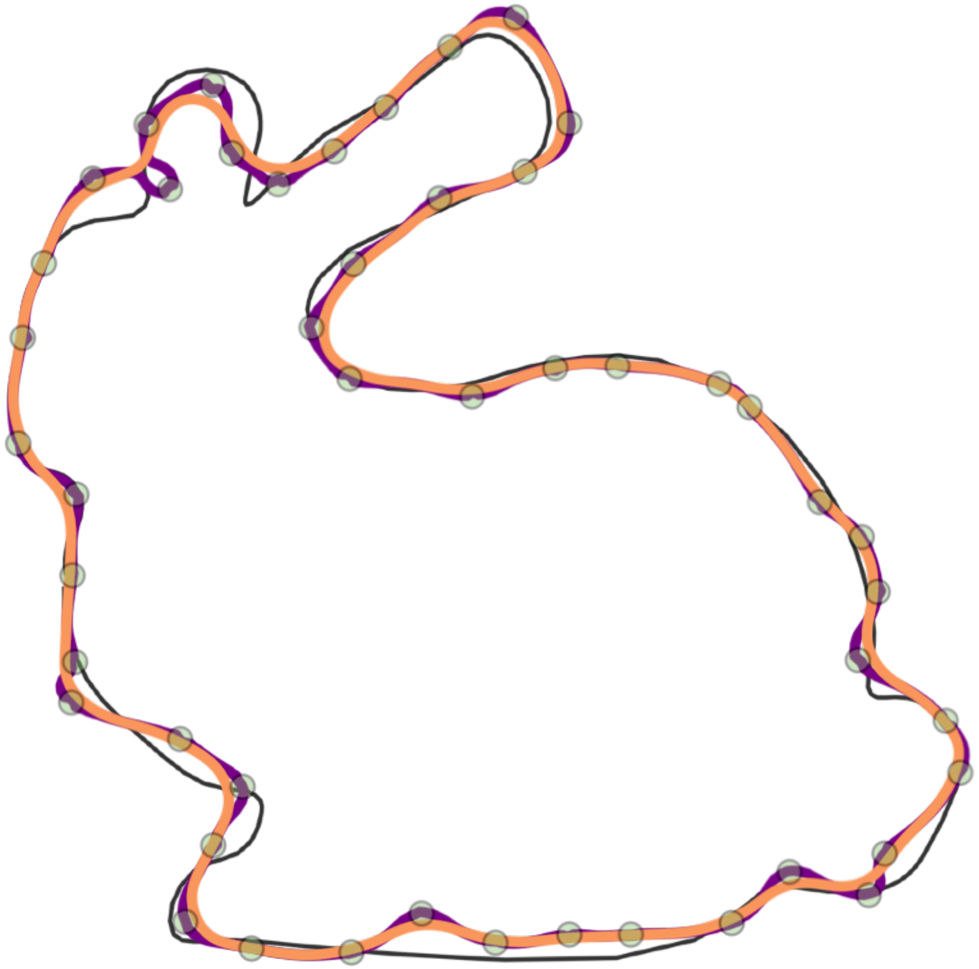
$$\sum P_{\varepsilon}\left(x, y_i, \hat{n}_i\right) \cdot f_i$$



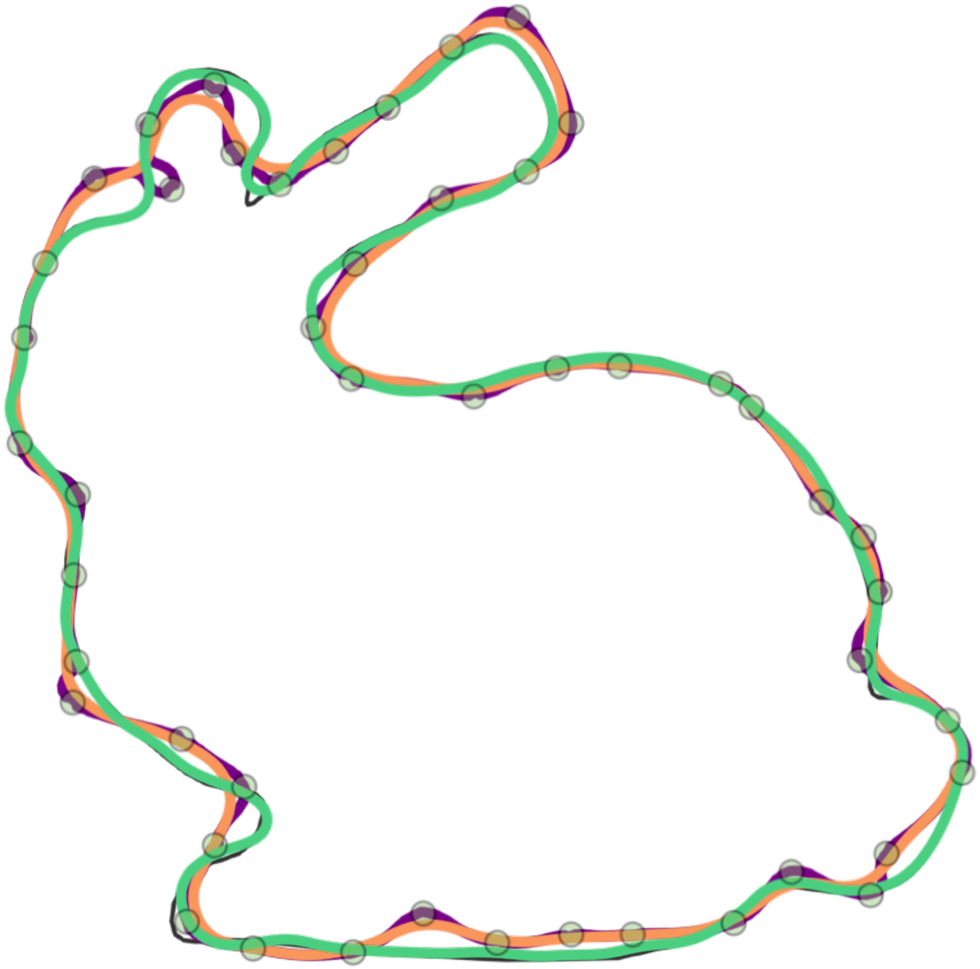




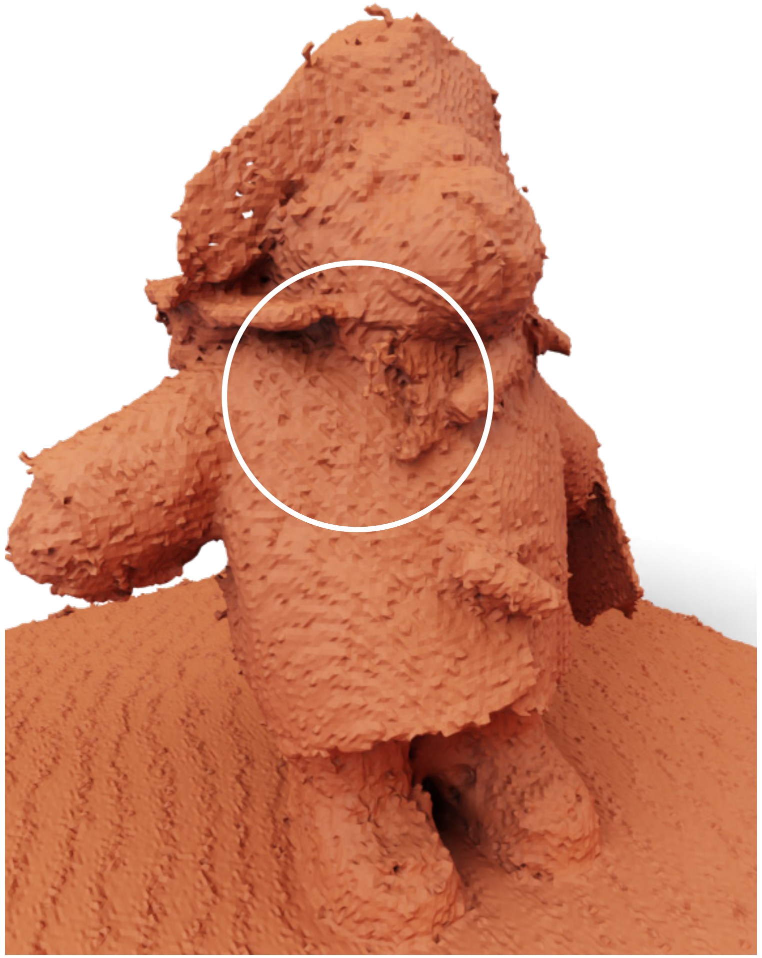
point-cloud winding number



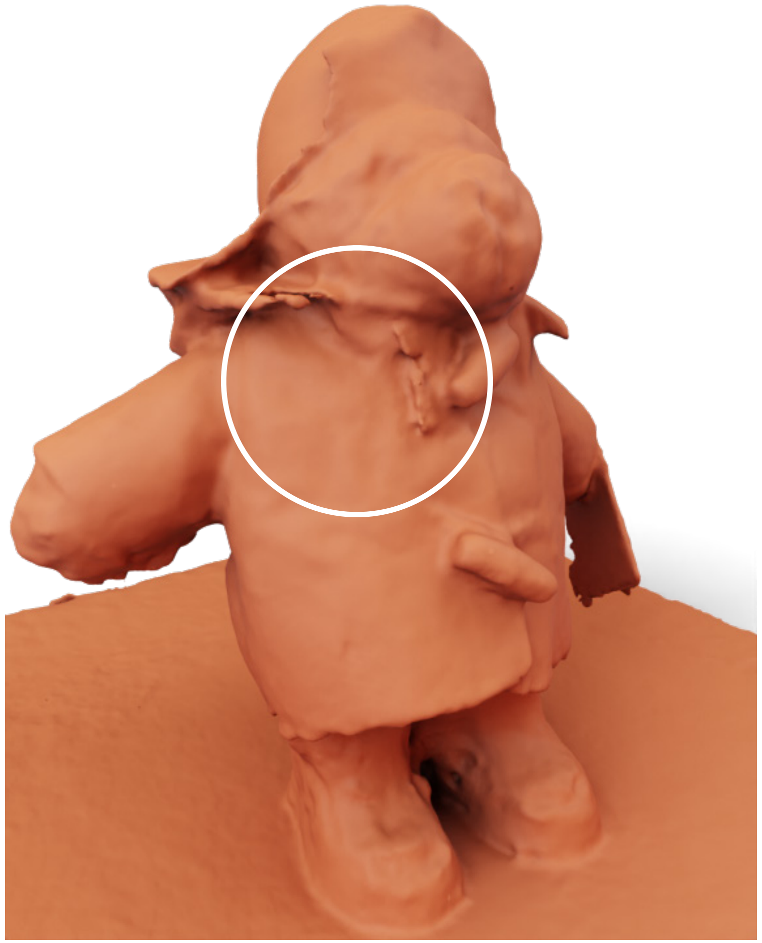
regularized winding number



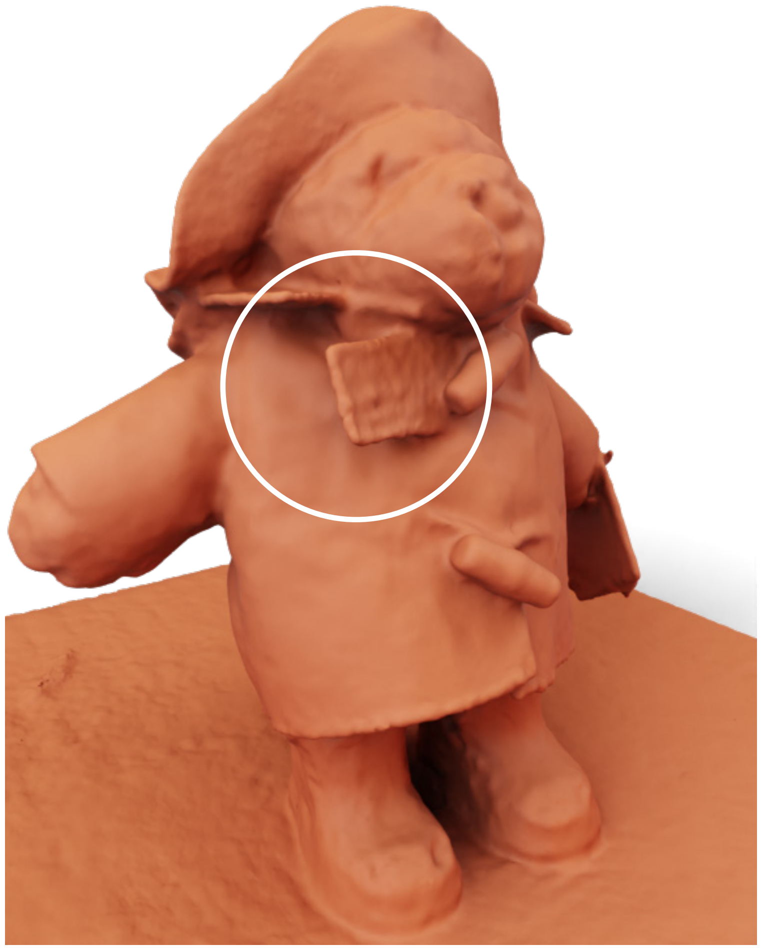
regularized dipole sum



winding number

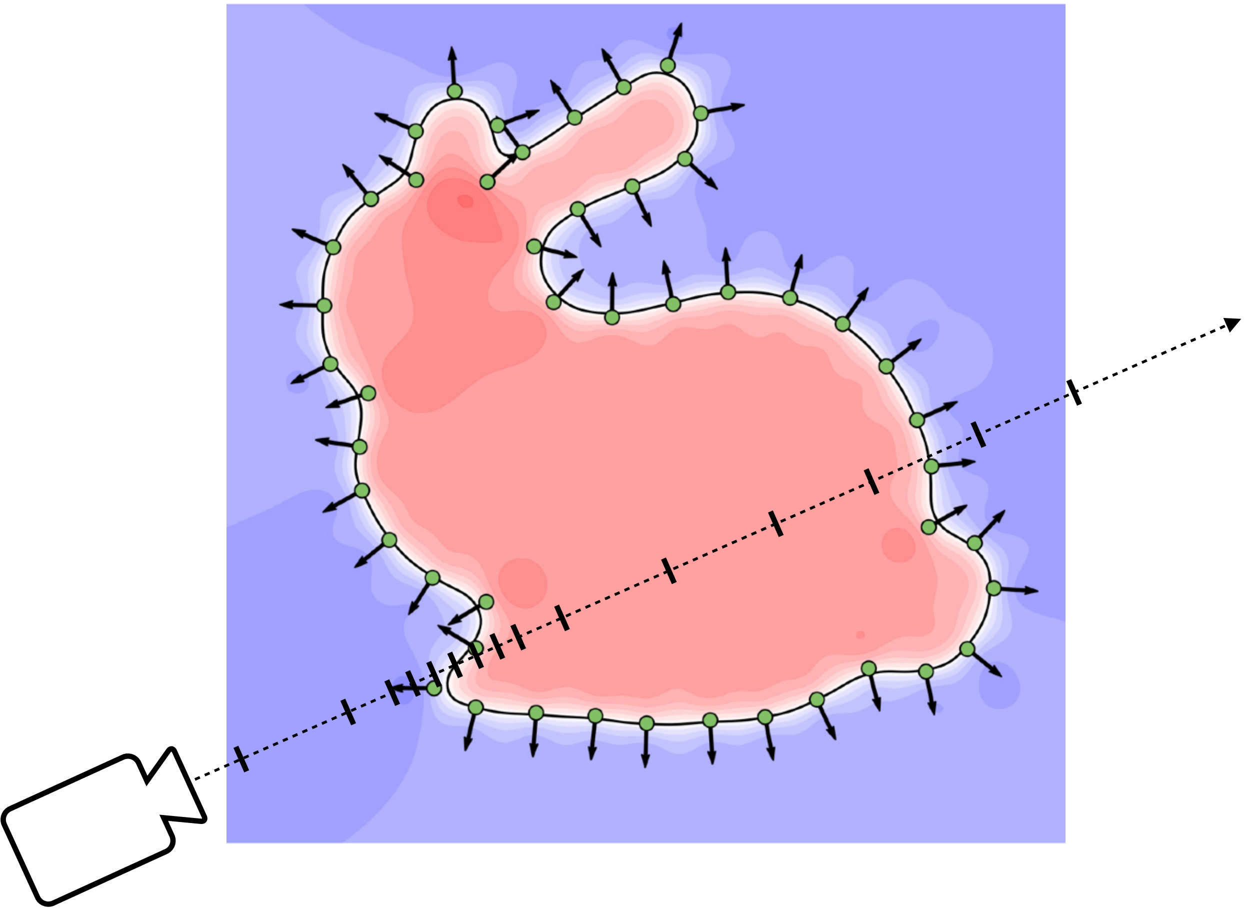


regularized
winding number

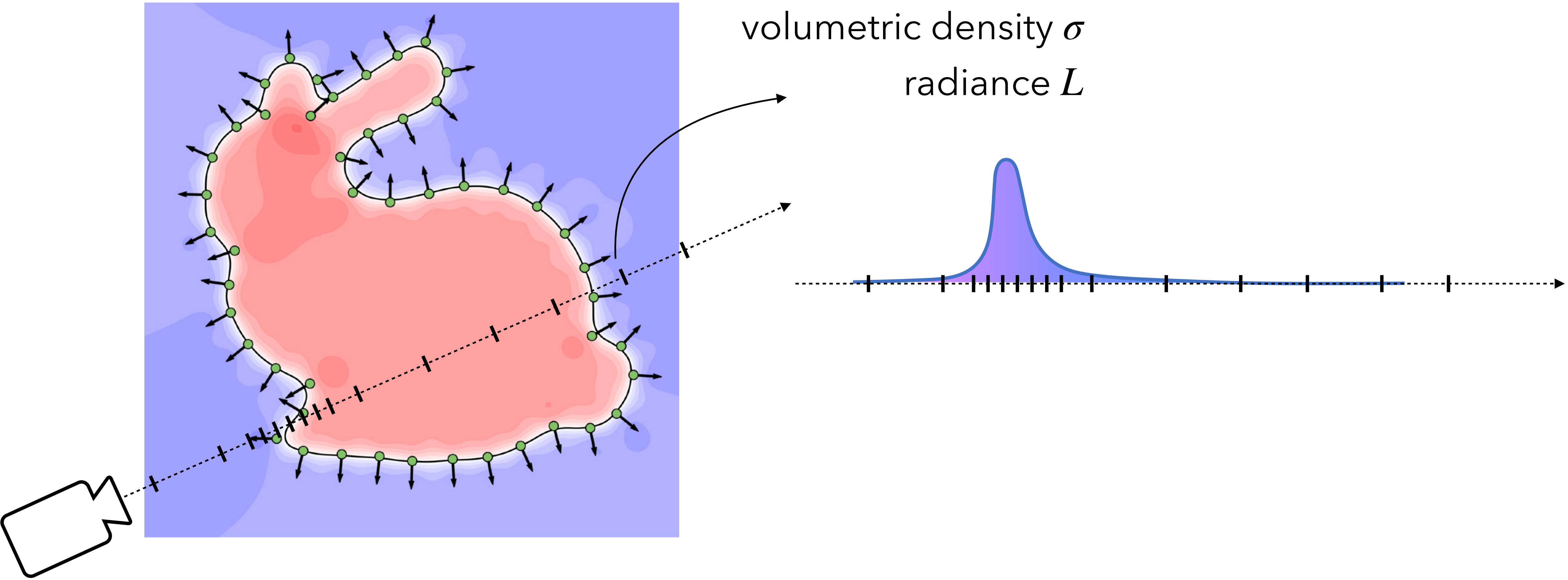


regularized
dipole sum

volume rendering



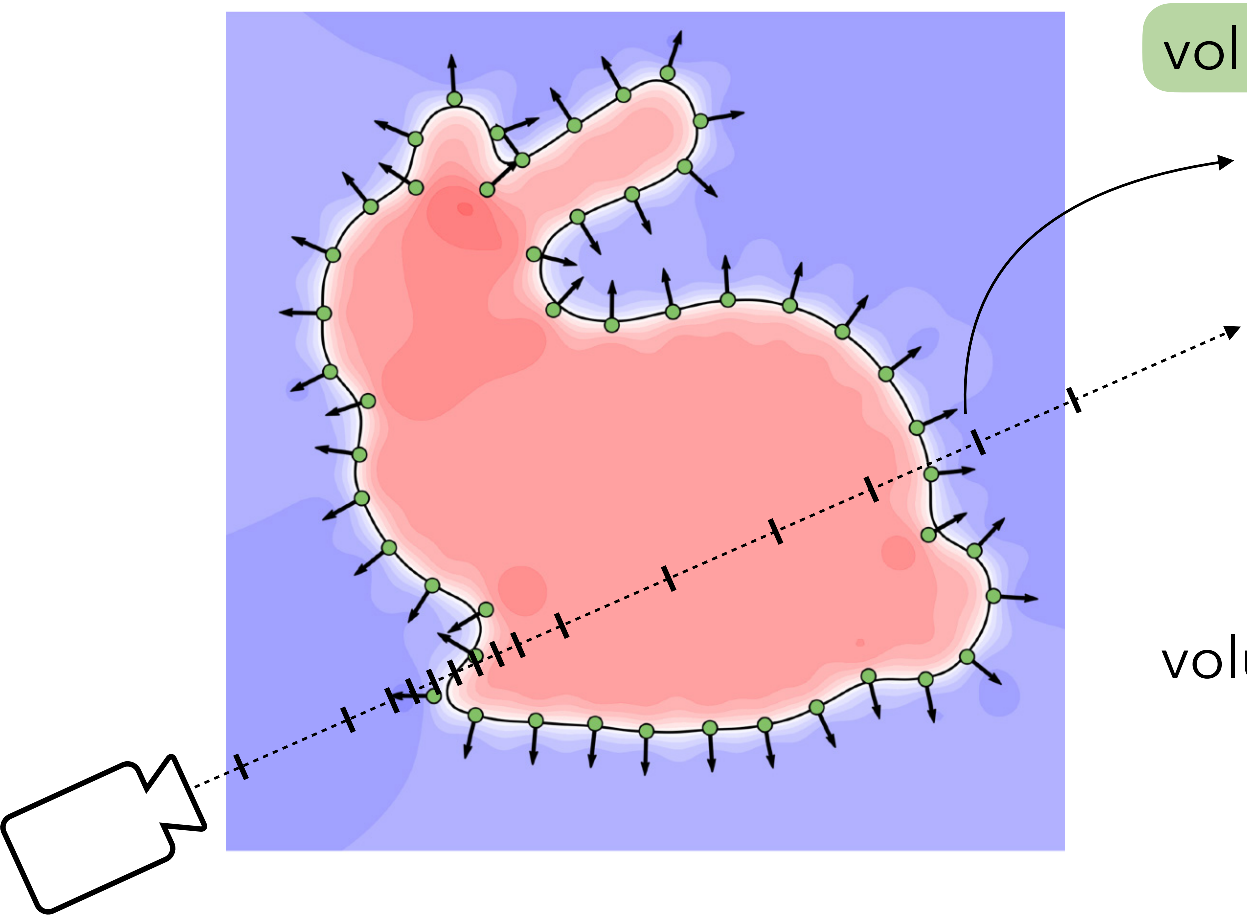
volume rendering



volumetric density σ

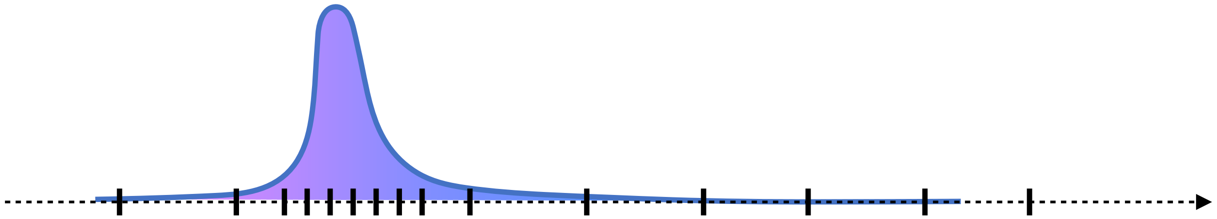
radiance L

volume rendering



volumetric density σ

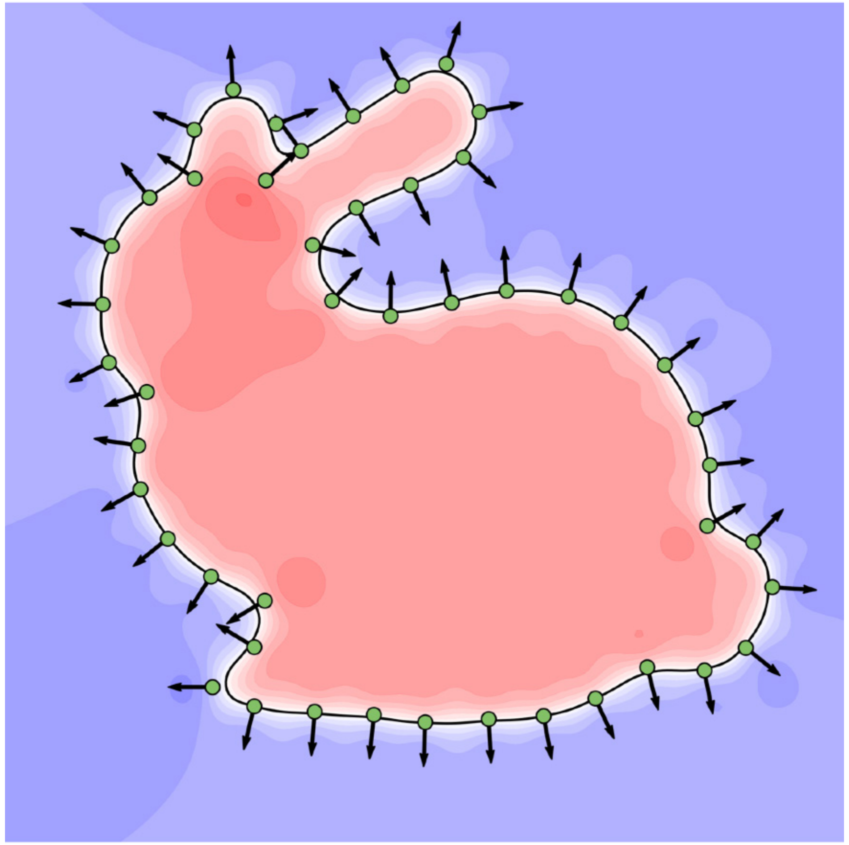
radiance L



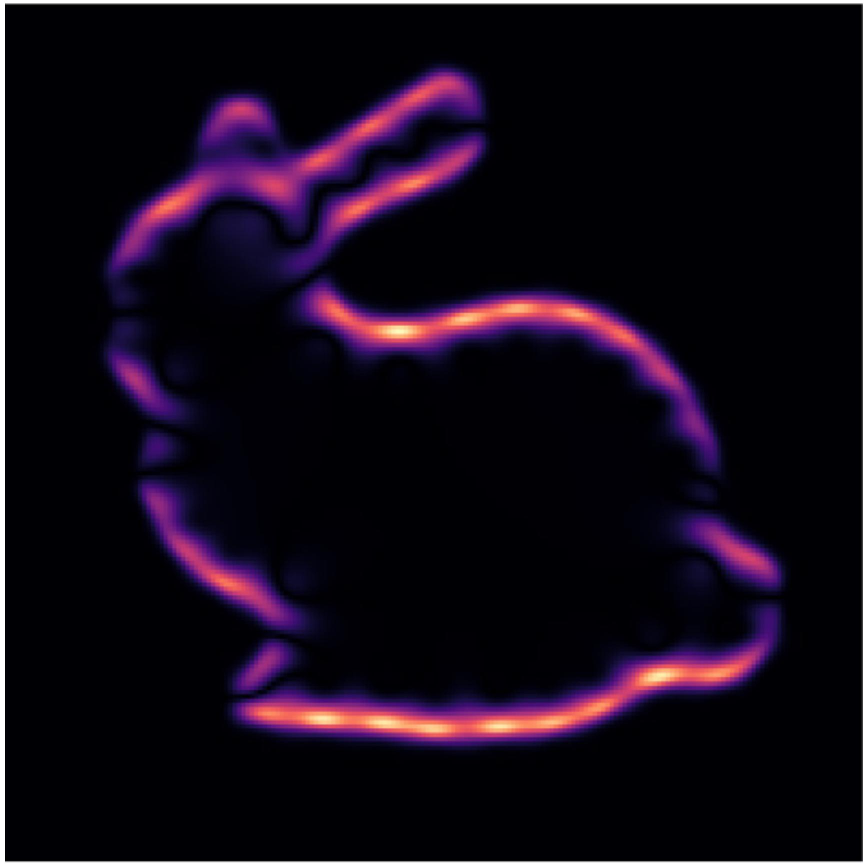
volume rendering equation:

$$c = \int L(x(t))\,\sigma(x(t))\,e^{\int \sigma(x(s))\,\mathrm{d}s}\,\mathrm{d}t$$

volume rendering



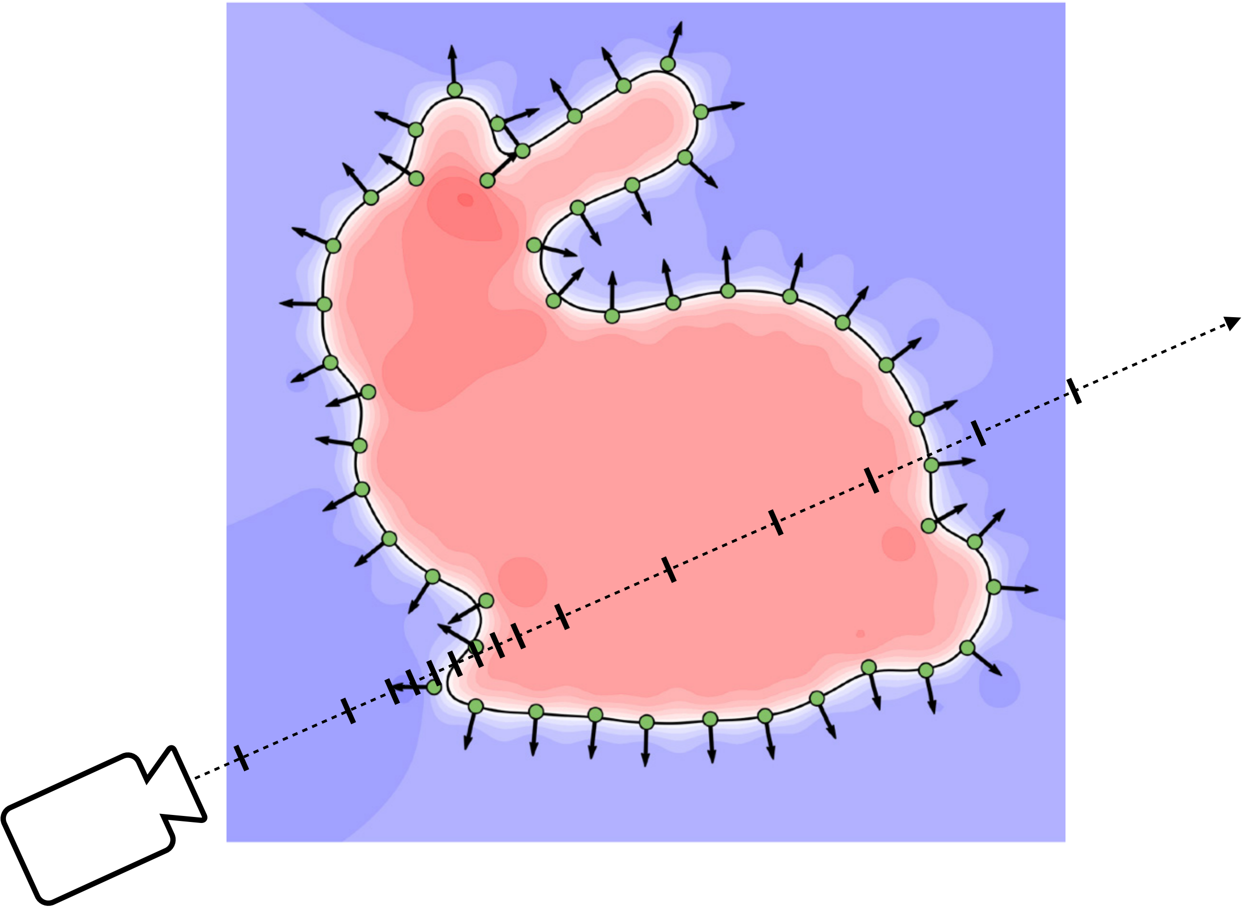
regularized
dipole sum



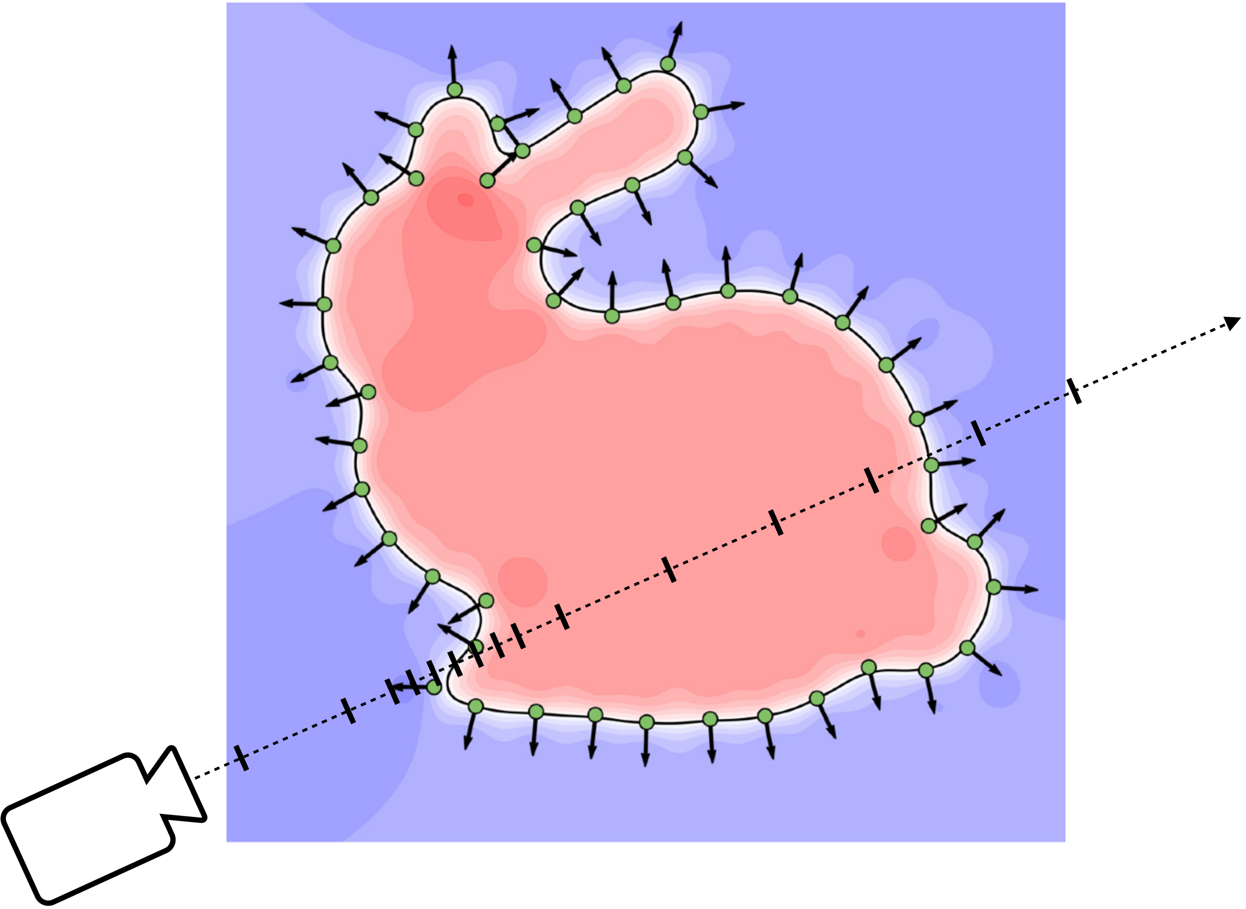
volumetric density σ

Miller et al., Objects as volumes: A stochastic geometry view of opaque solids, CVPR 2024

volume rendering



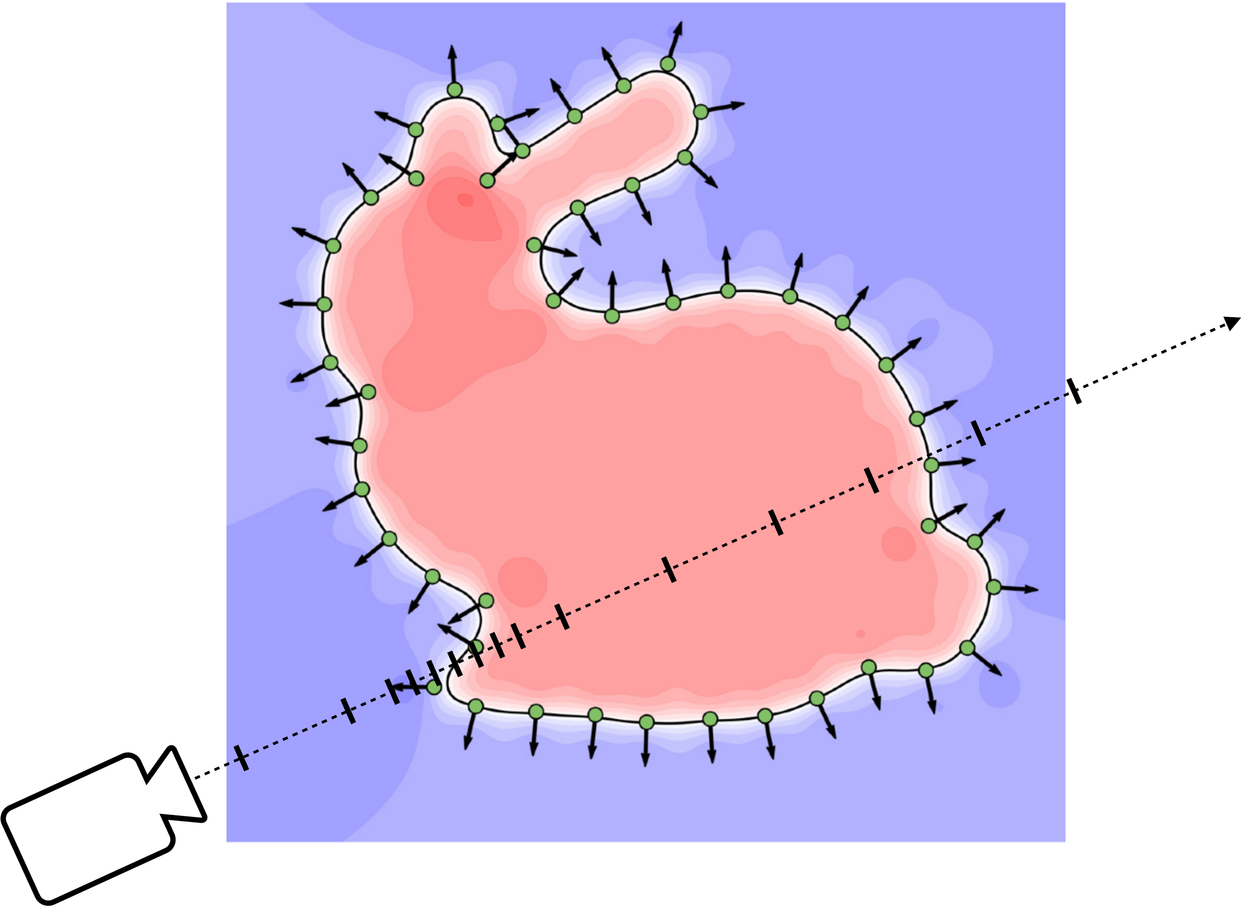
volume rendering



geometry

$$f(x) = \sum P_\varepsilon \left(x, y_i, \hat{n}_i\right) \cdot f_i$$

volume rendering



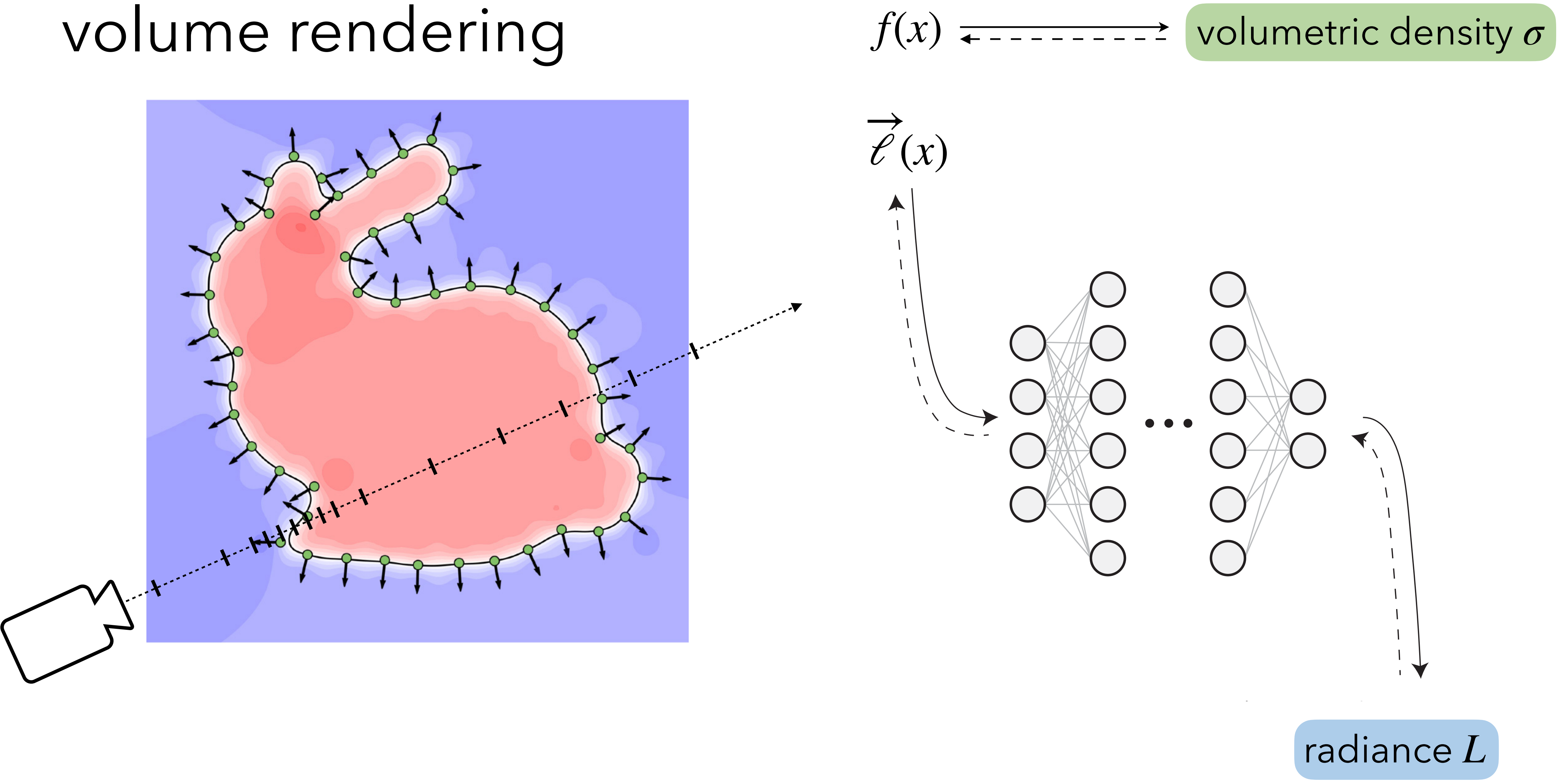
geometry

$$f(x) = \sum P_\varepsilon\left(x, y_i, \hat{n}_i\right) \cdot f_i$$

appearance

$$\vec{\ell}(x) = \sum P_\varepsilon\left(x, y_i, \hat{n}_i\right) \cdot \vec{\ell}_i$$

volume rendering



$f(x)$

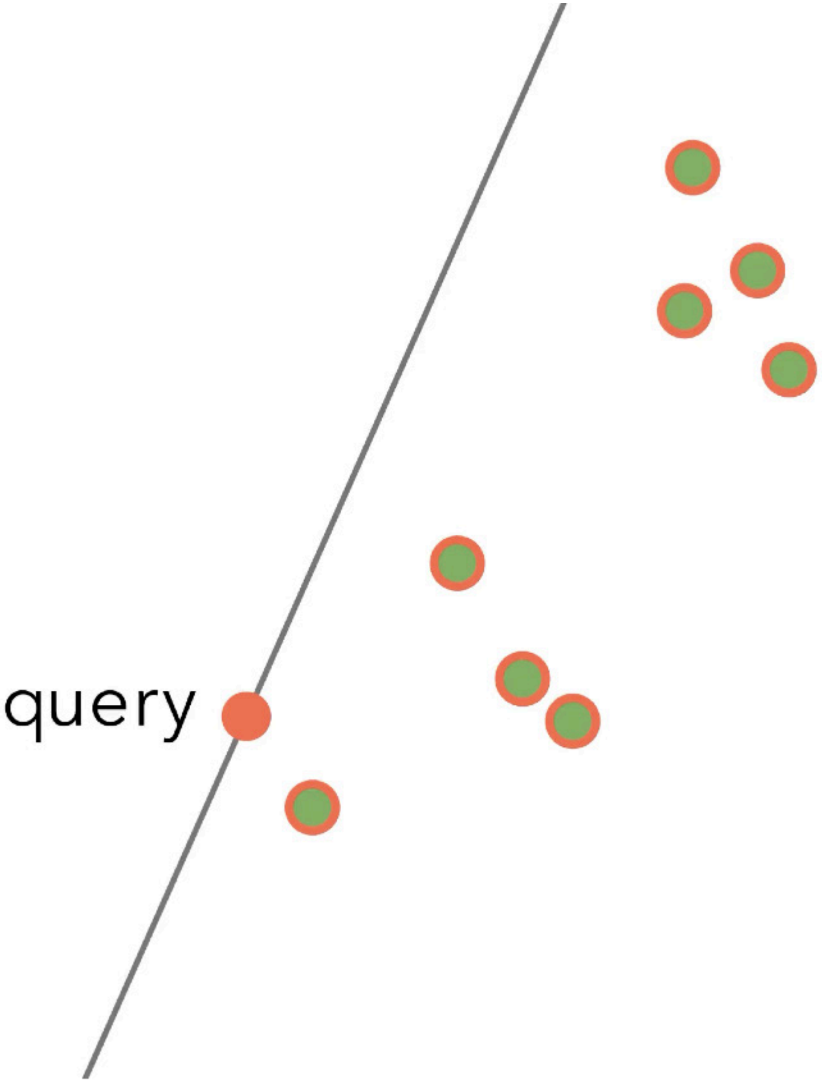
volumetric density σ

$\vec{\ell}(x)$

radiance L

naive summation over $\overset{\approx 100\text{ k}}{M}$ points for $\overset{\approx 50\text{ M}}{N}$ queries $\sum P_{\varepsilon}\left(x, y_i, \hat{n}_i\right) \cdot f_i$

$O(N \cdot M)$ time

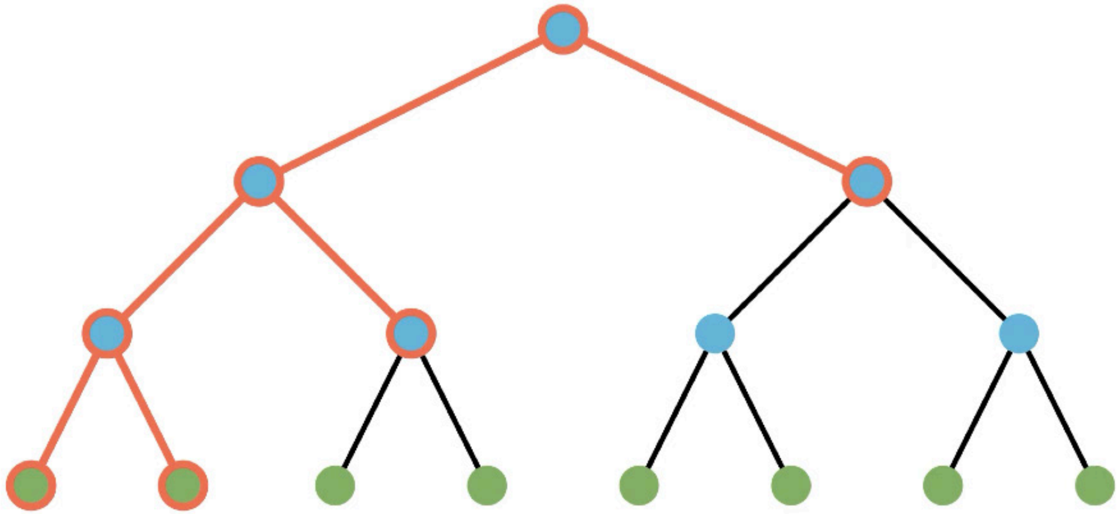


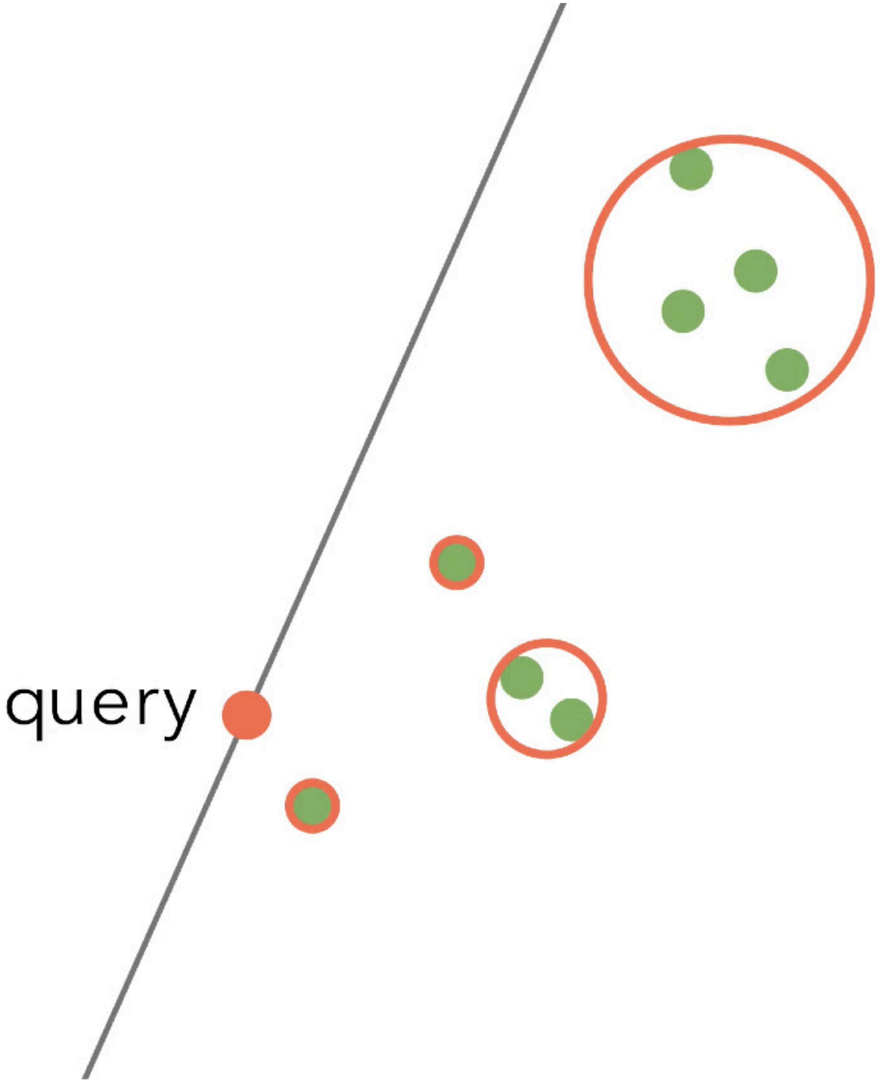
naive summation over $\overset{\approx 100\text{ k}}{M}$ points for $\overset{\approx 50\text{ M}}{N}$ queries $\sum P_\varepsilon\left(x, y_i, \hat{n}_i\right) \cdot f_i$

$O(N \cdot M)$ time

Barnes-Hut approximation

$O(N \cdot \log M)$ time



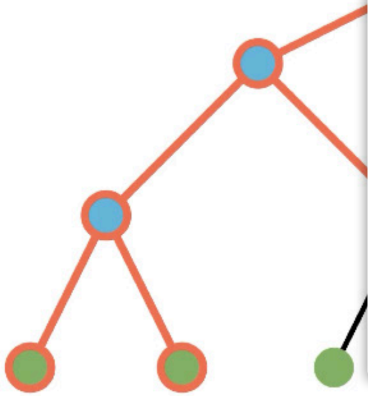


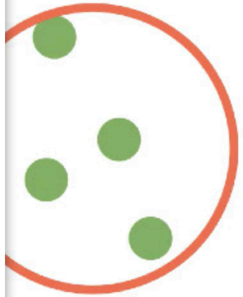
naive summ

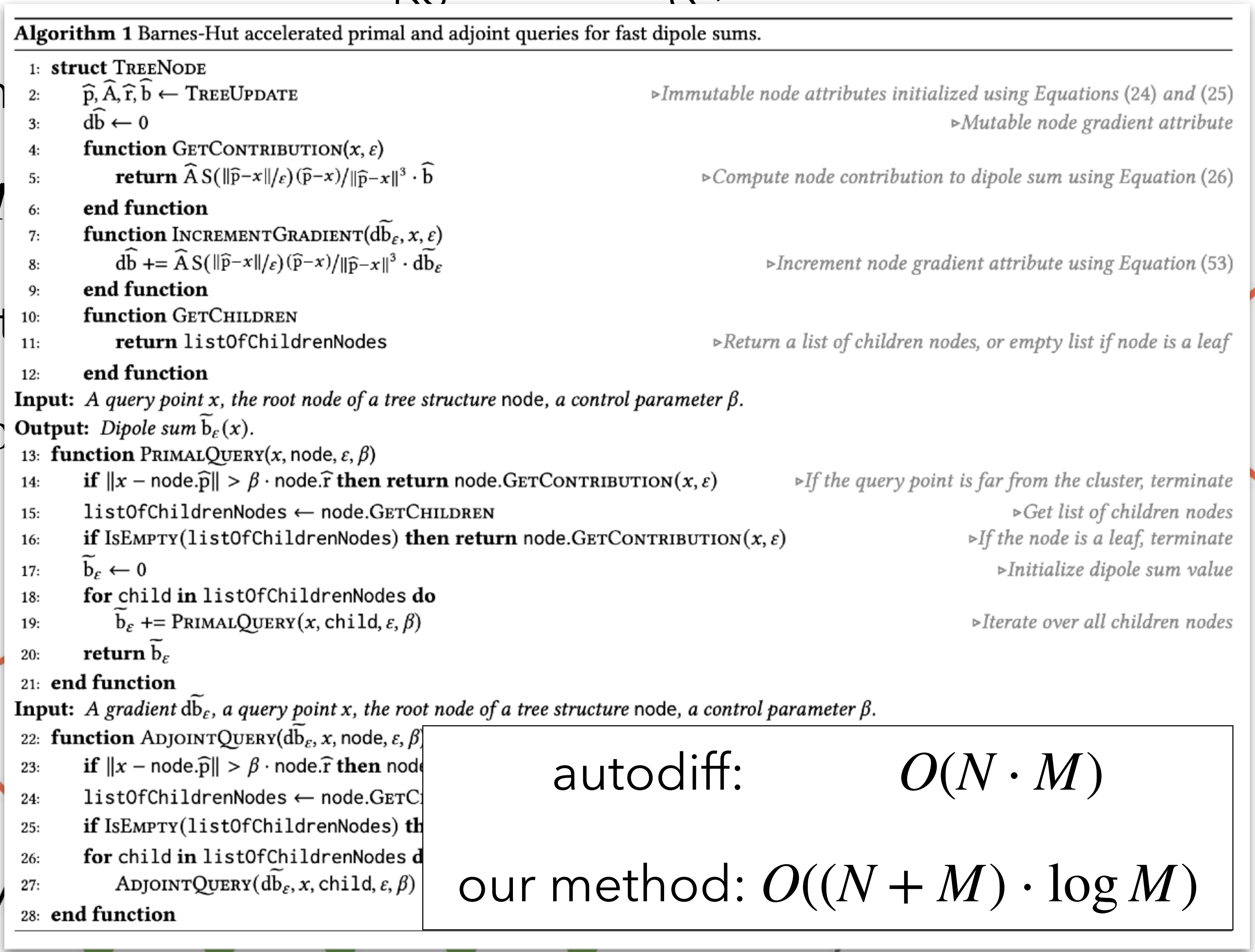
$O(N \cdot M$

Barnes-Hut

$O(N \cdot \mathrm{lo}$







Algorithm 1 Barnes-Hut accelerated primal and adjoint queries for fast dipole sums.

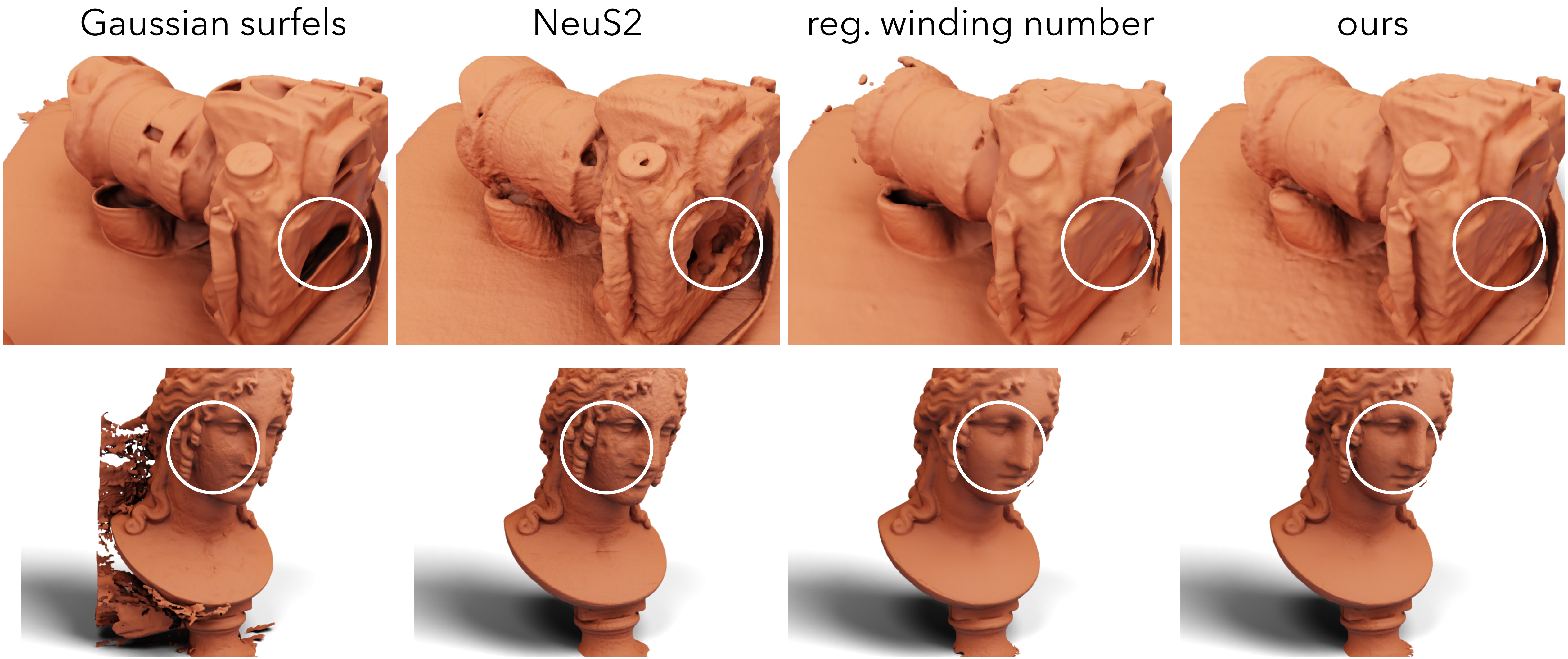
```
struct TreeNode
    $\widehat{p}, \widehat{A}, \widehat{r}, \widehat{b} \leftarrow$ TreeUpdate    ▷Immutable node attributes initialized using Equations (24) and (25)
    $d\widehat{b} \leftarrow 0$    ▷Mutable node gradient attribute
    function GetContribution($x, \varepsilon$)
        return $\widehat{A}\, S(\|\widehat{p}-x\|/\varepsilon)(\widehat{p}-x)/\|\widehat{p}-x\|^3 \cdot \widehat{b}$    ▷Compute node contribution to dipole sum using Equation (26)
    end function
    function IncrementGradient($d\widetilde{b}_\varepsilon, x, \varepsilon$)
        $d\widehat{b}$ += $\widehat{A}\, S(\|\widehat{p}-x\|/\varepsilon)(\widehat{p}-x)/\|\widehat{p}-x\|^3 \cdot d\widetilde{b}_\varepsilon$    ▷Increment node gradient attribute using Equation (53)
    end function
    function GetChildren
        return listOfChildrenNodes    ▷Return a list of children nodes, or empty list if node is a leaf
    end function
Input: A query point $x$, the root node of a tree structure node, a control parameter $\beta$.
Output: Dipole sum $\widetilde{b}_\varepsilon(x)$.
function PrimalQuery($x$, node, $\varepsilon, \beta$)
    if $\|x - \text{node}.\widehat{p}\| > \beta \cdot \text{node}.\widehat{r}$ then return node.GetContribution($x, \varepsilon$)    ▷If the query point is far from the cluster, terminate
    listOfChildrenNodes ← node.GetChildren    ▷Get list of children nodes
    if IsEmpty(listOfChildrenNodes) then return node.GetContribution($x, \varepsilon$)    ▷If the node is a leaf, terminate
    $\widetilde{b}_\varepsilon \leftarrow 0$    ▷Initialize dipole sum value
    for child in listOfChildrenNodes do
        $\widetilde{b}_\varepsilon$ += PrimalQuery($x$, child, $\varepsilon, \beta$)    ▷Iterate over all children nodes
    return $\widetilde{b}_\varepsilon$
end function
Input: A gradient $d\widetilde{b}_\varepsilon$, a query point $x$, the root node of a tree structure node, a control parameter $\beta$.
function AdjointQuery($d\widetilde{b}_\varepsilon, x$, node, $\varepsilon, \beta$)
    if $\|x - \text{node}.\widehat{p}\| > \beta \cdot \text{node}.\widehat{r}$ then nod
    listOfChildrenNodes ← node.GetC
    if IsEmpty(listOfChildrenNodes) th
    for child in listOfChildrenNodes d
        AdjointQuery($d\widetilde{b}_\varepsilon, x$, child, $\varepsilon, \beta$)
end function
```

autodiff: $O(N \cdot M)$

our method: $O((N+M) \cdot \log M)$

Blended MVS: reference

Blended MVS



Gaussian surfels

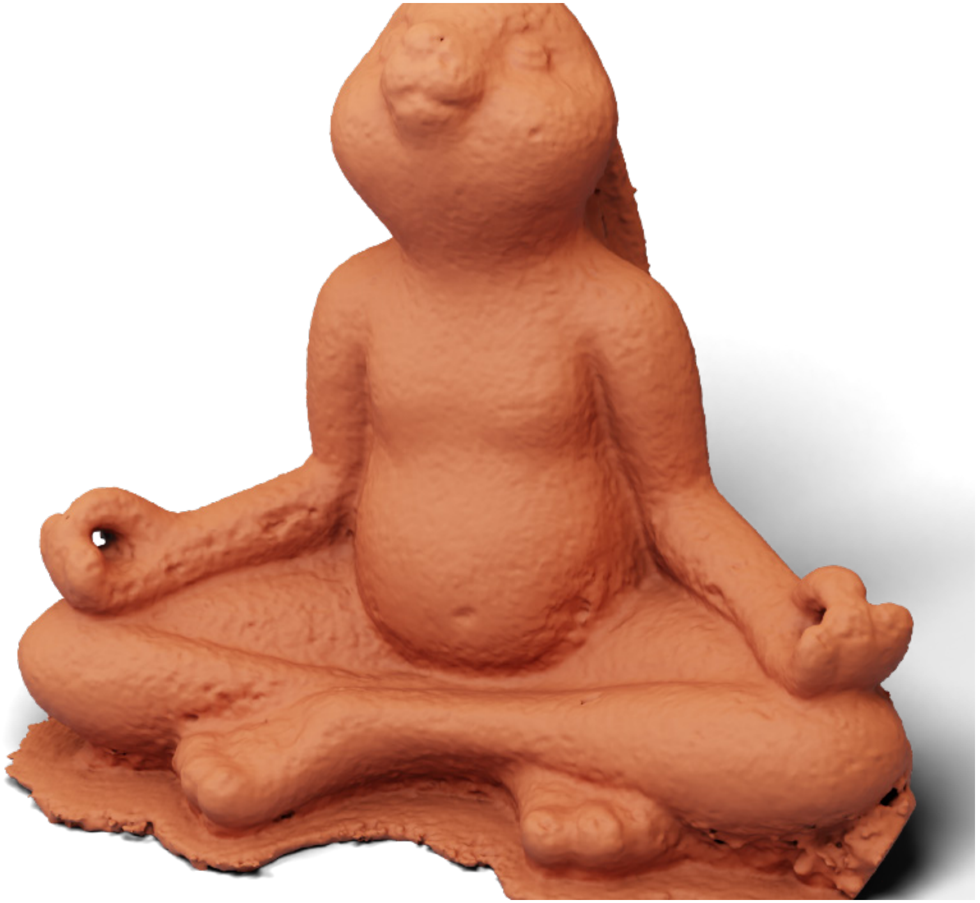
NeuS2

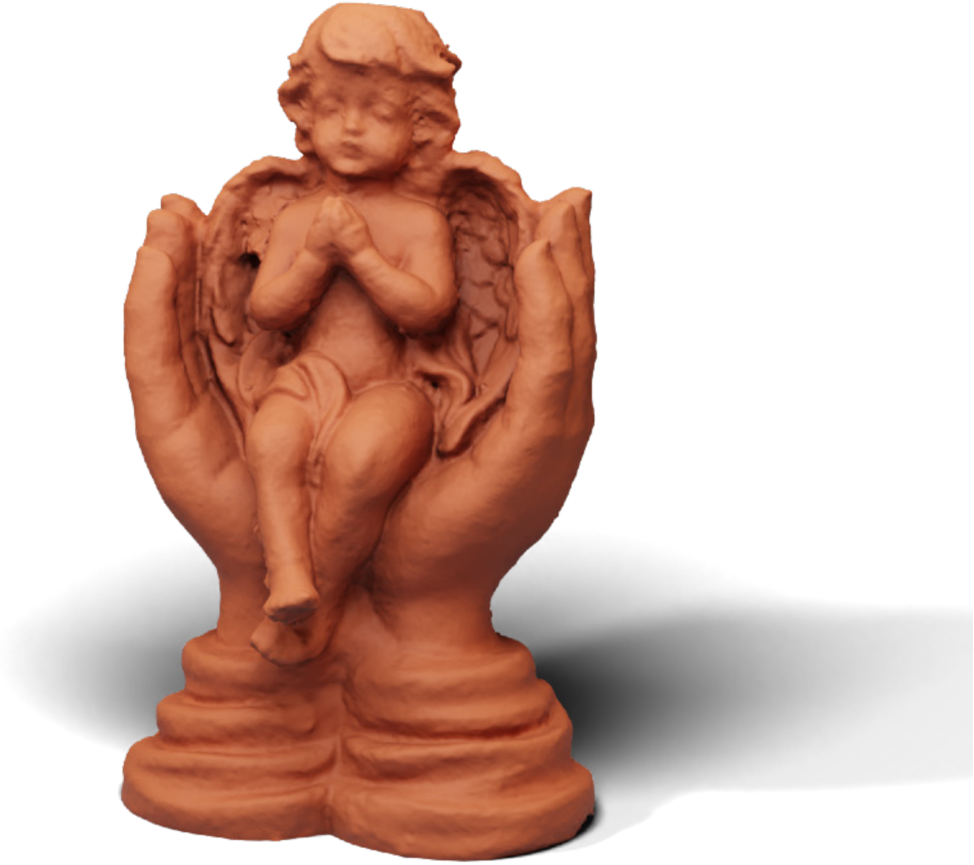
reg. winding number

ours

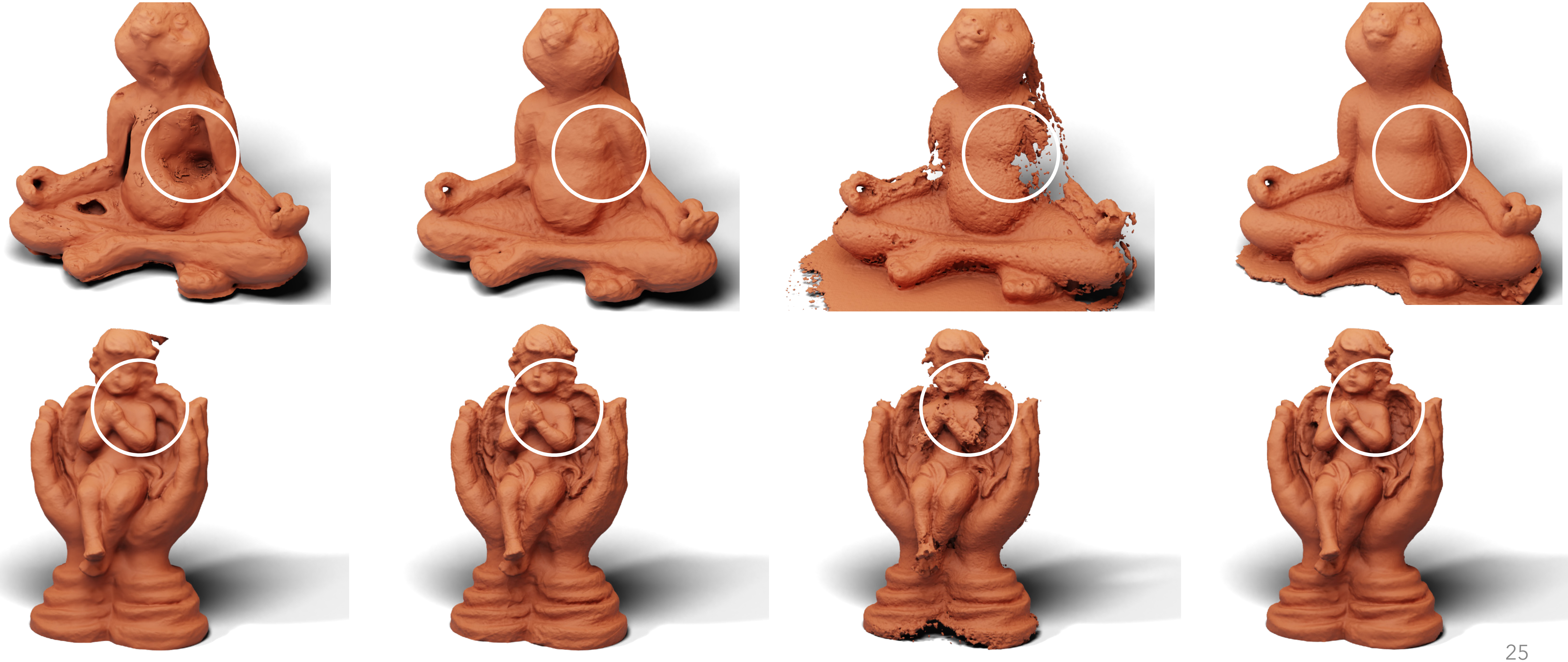
DTU: reference

DTU: ours





DTU



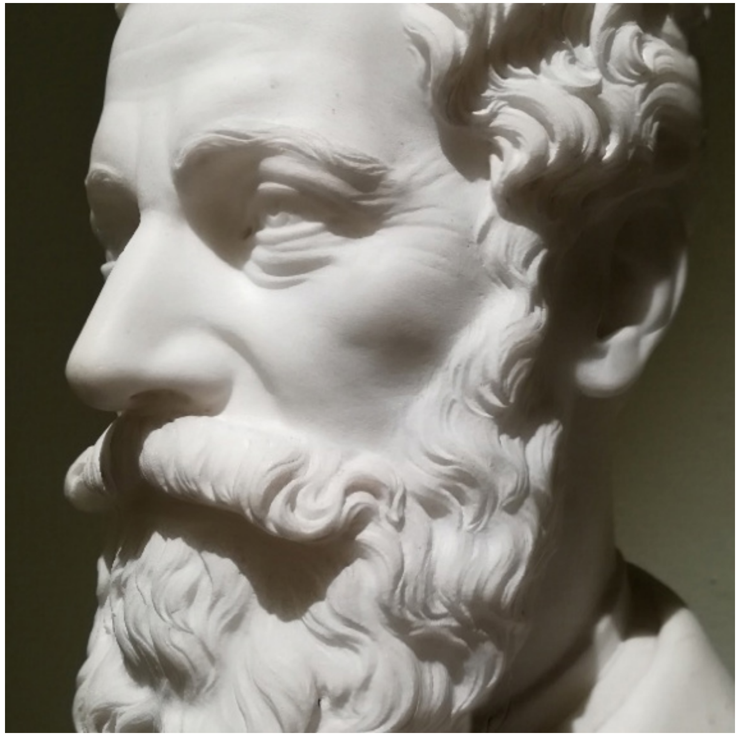
Gaussian surfels

NeuS2

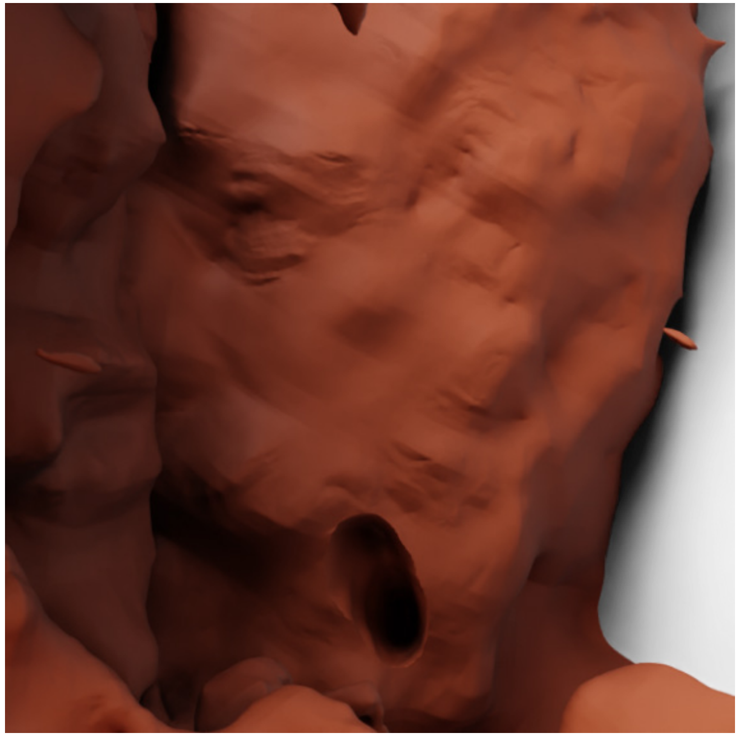
reg. winding number

ours

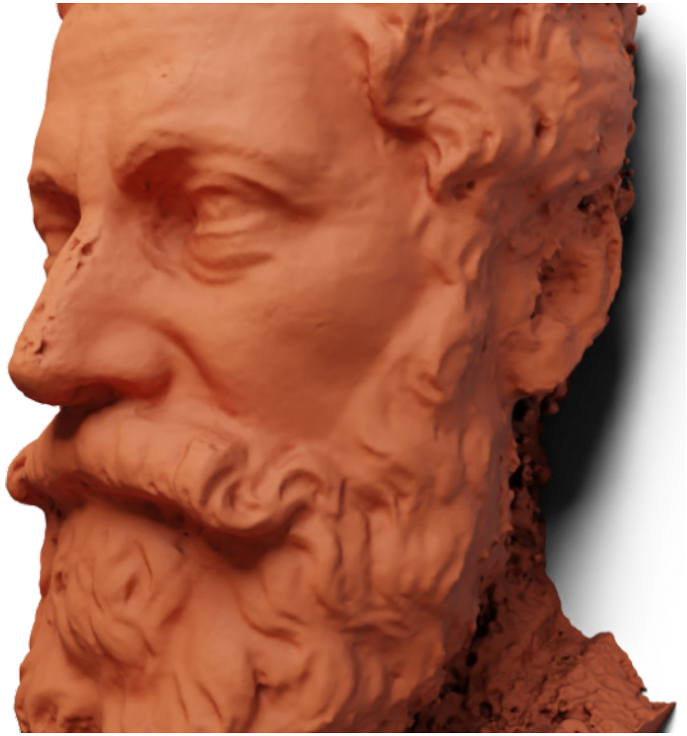
importance of geometric regularization



reference

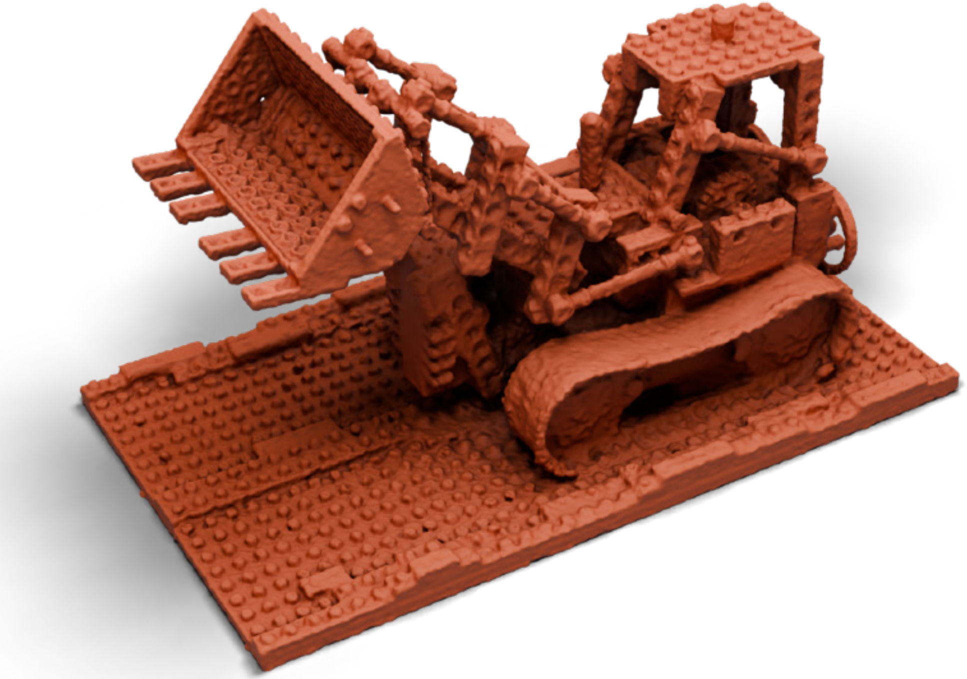


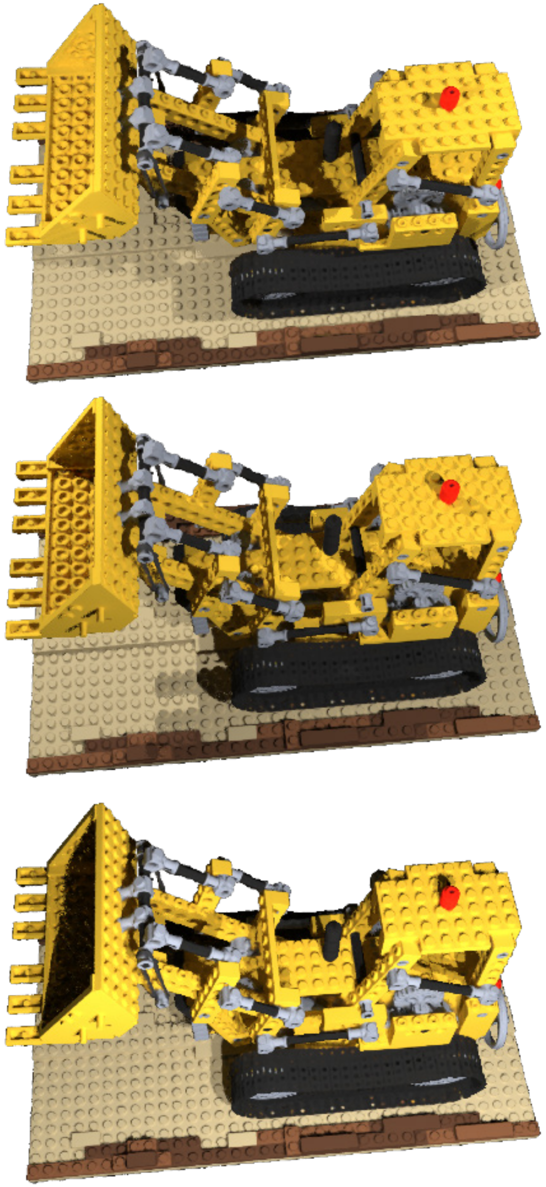
neuralangelo (14 hrs)



ours (10 mins)

extensive visualizations & additional results





code & data available on our website!

https://imaging.cs.cmu.edu/fast_dipole_sums/

This work was supported by NSF award 1900849, NSF Graduate Research Fellowship DGE2140739, an NVIDIA Graduate Fellowship for Miller, and a Sloan Research Fellowship for Gkioulekas.