Ray Tracing Harmonic Functions

MARK GILLESPIE, Carnegie Mellon University, USA
DENISE YANG, Carnegie Mellon University, USA and Pixar Animation Studios, USA
MARIO BOTSCH, TU Dortmund University, Germany
KEENAN CRANE, Carnegie Mellon University, USA

Sphere tracing is a fast and high-quality method for visualizing surfaces encoded by signed distance functions (SDFs). We introduce a similar method for a completely different class of surfaces encoded by *harmonic functions*, opening up rich new possibilities for visual computing. Our starting point is similar in spirit to sphere tracing: using conservative *Harnack bounds* on the growth of harmonic functions, we develop a *Harnack tracing* algorithm for visualizing level sets of harmonic functions, including those that are angle-valued and exhibit singularities. The method takes much larger steps than naïve ray marching, avoids numerical issues common to generic root finding methods and, like sphere tracing, needs only perform pointwise evaluation of the function at each step. For many use cases, the method is fast enough to run real time in a shader program. We use it to visualize smooth surfaces directly from point clouds (via Poisson surface reconstruction) or polygon soup (via generalized winding numbers) without linear solves or mesh extraction. We also use it to visualize nonplanar polygons (possibly with holes), surfaces from architectural geometry, mesh "exoskeletons", and key mathematical objects including knots, links, spherical harmonics, and Riemann surfaces. Finally we show that, at least in theory, Harnack tracing provides an alternative mechanism for visualizing arbitrary implicit surfaces.

CCS Concepts: • **Computing methodologies** → **Ray tracing**; *Shape analysis*; • **Mathematics of computing** → *Numerical analysis.*

Additional Key Words and Phrases: Ray tracing, sphere tracing, implicit surfaces, harmonic function, Harnack inequality

ACM Reference Format:
Mark Gillespie, Denise Yang, Mario Botsch, and Keenan Crane. 2024. Ray Tracing Harmonic Functions. *ACM Trans. Graph.* 43, 4, Article 99 (July 2024), 18 pages. https://doi.org/10.1145/3658201

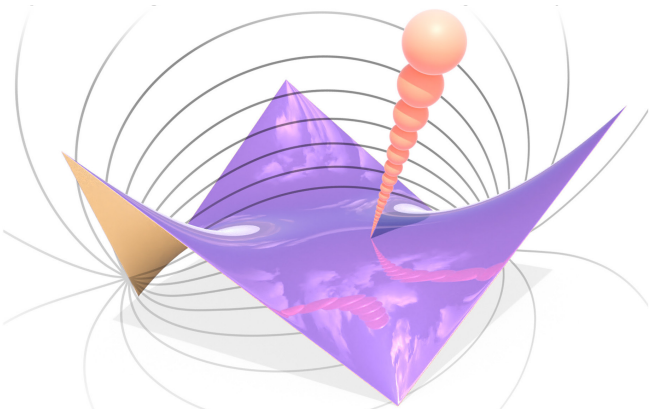
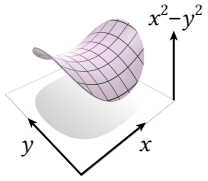


Fig. 1. We introduce a ray tracing algorithm for a novel class of surfaces defined by level sets of harmonic functions. Here for instance we directly visualize a nonplanar polygon which has no well-defined inside or outside—and hence cannot be represented by an ordinary implicit function or SDF. Isolines depict a 2D slice of the harmonic function; spheres show conservative *Harnack bounds* along a ray. Note the smooth reflection lines, even near edges and vertices where the function is highly singular.

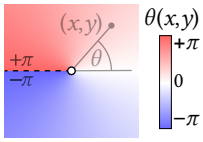
1 INTRODUCTION

A function $f : \Omega \to \mathbb{R}$ on a domain $\Omega \subset \mathbb{R}^n$ is *harmonic* if it satisfies the Laplace equation

$$\Delta f(\mathbf{x}) = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2} f(\mathbf{x}) = 0 \qquad (1)$$



at all points $\mathbf{x} \in \Omega$. The function $f(x, y) = x^2 - y^2$ is harmonic on $\mathbb{R}^2$, and exhibits the saddle shape characteristic of harmonic functions (see inset). Harmonic functions can be *angle-valued* (Section 3.2), and can exhibit singularities. The function $\theta(x, y) = \text{atan2}(y, x)$ in the inset is a model example: it jumps by 2π at $y = 0$, and is singular at $x = y = 0$. Considering angle-valued functions allow us to represent implicit surfaces with boundary (Figure 1).



Despite their special form, harmonic functions arise naturally in many contexts. From a variational perspective, they describe the smoothest function that agrees with given observations—providing a powerful tool for interpolating data over geometric domains (*e.g.*, color [Orzan et al. 2008] or displacement [Joshi et al. 2007]). From a spectral perspective, Fourier basis functions on $\mathbb{R}^n$ can be extended to harmonic functions on $\mathbb{R}^{n+1}$, allowing any function to be well-approximated by a "slice" of a harmonic one (as explored in Section 4.7.1). Finally, from an integral perspective, harmonic functions are characterized by the fact that $f(\mathbf{x})$ is equal to the mean value of f over any ball around $\mathbf{x}$—connecting them to steady-state solutions for a vast array of physical equations (electrostatics, gravitation, heat transfer, *etc.*). This so-called mean value property is the starting point for the *Harnack inequality* used in our algorithm to bound the change in the value of a positive harmonic function (Section 2). Although Harnack inequalities are widely applied in the mathematical analysis of partial differential equations, they have not (to our knowledge) been applied to rendering or image synthesis.

Authors' addresses: Mark Gillespie, Carnegie Mellon University, USA, mgillesp@cs.cmu.edu; Denise Yang, Carnegie Mellon University, USA, Pixar Animation Studios, USA, deniseyg29@gmail.com; Mario Botsch, TU Dortmund University, Germany, mario.botsch@tu-dortmund.de; Keenan Crane, Carnegie Mellon University, USA, kmcrane@cs.cmu.edu.

0730-0301/2024/7-ART99
https://doi.org/10.1145/3658201

In this paper, we are interested in ray tracing *level sets* of a harmonic function f, *i.e.*, sets

$$\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) = f^*\}, \tag{2}$$

where f^* is a fixed target value. In particular, given a ray

$$\mathbf{r}(t) = \mathbf{r}_0 + t\mathbf{v} \tag{3}$$

starting at $\mathbf{r}_0$ and extending in the direction $\mathbf{v}$, we seek the first time $t > 0$ such that $f(\mathbf{r}(t)) = f^*$. Unfortunately, general-purpose intersection strategies (detailed in Section 5) do not provide a robust solution: they (i) provide no first-hit guarantee, (ii) do a poor job near singularities, and (iii) do not handle angle-valued functions. To date, ray intersection algorithms provide guarantees only for a few special classes of surfaces, enumerated in Section 5.2. Our method adds one more entry to this list: surfaces defined by harmonic functions. More precisely, it enables reliable ray tracing of

- level sets of harmonic functions (possibly with singularities),
- lower-dimensional slices through harmonic level sets, and
- height fields of harmonic functions (possibly multivalued).

Our method is inspired by the *sphere tracing* algorithm of Hart [1996], which ray traces level sets of a function f with a known *Lipschitz bound*, *i.e.*, a value $C > 0$ such that, for all points $\mathbf{x}, \mathbf{y}$ we have $|f(\mathbf{x}) - f(\mathbf{y})| \leq C\|\mathbf{x} - \mathbf{y}\|$. At any time t, one can hence be certain that no point of the surface $\mathcal{S}$ is contained within a ball of radius $R = (f(\mathbf{r}(t)) - f^*)/C$—providing a conservative step size for ray tracing. An important special case are *signed distance functions (SDFs)*, with Lipschitz constant $C = 1$. Sphere tracing was originally motivated by ray tracing fractal *quaternion Julia sets* [Hart et al. 1989], which admit closed-form distance bounds; in more recent times, it has become a basic strategy for evaluating neural implicit surfaces [Takikawa et al. 2021, Section 2]. Just as in sphere tracing, we truncate the ray by the largest "safe" sphere—the key difference is that this safe radius is determined by reasoning about harmonic functions, rather than a Lipschitz function.

1.1 Outline

Section 2 gives some brief background on the Harnack inequality, used to define our *Harnack tracing* algorithm in Section 3. Section 4 walks through a variety of examples, including derivation of some bounds and formulas needed to apply our algorithm to several classes of surfaces. Here we also perform numerical evaluation of our method. We defer a discussion of related work to Section 5, in order to simultaneously make numerical comparisons to our method. Limitations and future work are discussed in Section 6.

2 THE HARNACK INEQUALITY

Let $f : B_R(\mathbf{x}_0) \to \mathbb{R}_{\geq 0}$ be a positive harmonic function on the open ball $B_R(\mathbf{x}_0)$ of radius $R > 0$ centered at a point $\mathbf{x}_0 \in \mathbb{R}^n$. The *Harnack inequality*, illustrated in Figure 2, provides a conservative upper and lower bound on the value of f for every point $\mathbf{x} \in B_R(\mathbf{x}_0)$ in terms of the distance $\rho := \|\mathbf{x} - \mathbf{x}_0\| < R$ and the value of f at $\mathbf{x}_0$:

$$\frac{(R-\rho)R^{n-2}}{(R+\rho)^{n-1}} f(\mathbf{x}_0) \leq f(\mathbf{x}) \leq \frac{(R+\rho)R^{n-2}}{(R-\rho)^{n-1}} f(\mathbf{x}_0) \tag{4}$$

(see Harnack [1887, §19] or Axler et al. [2013, §3.4]). Note in particular that the upper bound goes to $+\infty$ as $\rho \to R$, reflecting the fact

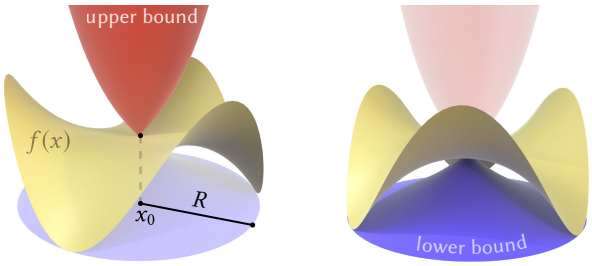


Fig. 2. For any positive harmonic function $f(\mathbf{x})$ on a ball of radius R, the Harnack inequality gives an upper and lower bound on the values of this function, purely in terms of R and the value $f(\mathbf{x}_0)$ at the ball center.

that a harmonic function can have arbitrarily large values along the boundary of the ball. Likewise, the lower bound goes to zero as $\rho \to R$, since a positive function can go to zero at any boundary point. The reason f must be *positive* is that there is no bound on the growth of a general harmonic function within a ball depending only on R and $f(\mathbf{x}_0)$. For instance, a linear function on $B_R(\mathbf{x}_0)$ can have arbitrarily large slope, but forcing a linear function to be positive ensures that its slope is no greater than $f(\mathbf{x}_0)/R$. Although the Harnack inequality applies only to positive harmonic functions, we can still use it to build algorithms that ray trace more general harmonic functions, as discussed in Section 3.1.

2.1 Largest Step Size

We will rewrite the Harnack inequality in a form that makes it more directly useful for our algorithms. In particular, suppose we are standing at a point $\mathbf{x}_0 \in \mathbb{R}^3$ and want to know the maximum step size ρ_{lower} we can take in any direction $\mathbf{v}$ such that $f(\mathbf{x}_0 + \rho\mathbf{v})$ is no smaller than a given lower bound $f_- \in (0, f(\mathbf{x}_0))$. In 3D, we have

$$\rho_{\text{lower}} := \frac{R}{2}\left(-(2 + a_-) + \sqrt{a_-^2 + 8a_-}\right), \tag{5}$$

where $a_- := f(\mathbf{x}_0)/f_-$. Likewise, if we want to avoid exceeding an upper bound $f^+ \in (f(\mathbf{x}_0), \infty)$, then we have a maximum step size

$$\rho_{\text{upper}} := \frac{R}{2}\left(a_+ + 2 - \sqrt{a_+^2 + 8a_+}\right), \tag{6}$$

where $a_+ := f(\mathbf{x}_0)/f_+$. Hence, if our goal is to intersect a level set with value f^*, we can simply compute a single step size

$$\rho := \frac{R}{2}\left|a + 2 - \sqrt{a^2 + 8a}\right|, \tag{7}$$

where $a := f(\mathbf{x}_0)/f^*$. This bound provides a conservative (*i.e.*, "safe") step size, whether $f(\mathbf{x}_0)$ is above or below f^*.

3 ALGORITHM

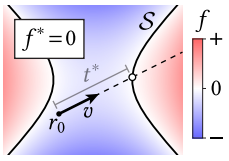
Our algorithm—which we call *Harnack tracing*—finds the first point along a ray intersecting the level set of a given harmonic function. Unlike the Harnack inequality from Section 2, we do not require that this function be positive. Instead, we require only that its value can be bounded from below within some ball around any given point. We first describe the algorithm in the case where f is smooth everywhere; later we will consider cases where f exhibits jumps and singularities (Section 3.2), which are important for applications.

3.1 Harnack Tracing

Let $f : \mathbb{R}^3 \to \mathbb{R}$ be a harmonic function, and let $\mathbf{r}(t)$ be the ray

$$\mathbf{r}(t) \coloneqq \mathbf{r}_0 + t\mathbf{v}$$

starting at $\mathbf{r}_0 \in \mathbb{R}^3$ and moving in direction $\mathbf{v} \in \mathbb{R}^3$ (see inset). For a target value f^*, the Harnack tracing algorithm computes the smallest time $t^* > 0$ such that



$$f(\mathbf{r}(t^*)) = f^*,$$

i.e., the time where the ray first pierces the surface

$$\mathcal{S} \coloneqq f^{-1}(f^*) = \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) = f^*\}.$$

To do so, we use the inequalities from Section 2.1 to find the largest step size ρ for which the ray is guaranteed not to pass through $\mathcal{S}$. However, since these inequalities apply only to *positive* functions, we effectively "shift" f within a local ball to get a safe step size.

More explicitly, suppose that for any point $\mathbf{x} \in \mathbb{R}^3$, we have a radius $R > 0$ and value $c \in \mathbb{R}$ such that $f(\mathbf{y}) > c$ for all points $\mathbf{y} \in B_R(\mathbf{x})$. To find a step size ρ at some time t, we evaluate Equation 7, but using the shifted function value $f(\mathbf{r}(t)) - c$ and shifted target value $f^* - c$. We then increment t by ρ and repeat this process until $f(\mathbf{r}(t))$ is sufficiently close to f^*, or t exceeds some maximum time $t_{\max}$. Algorithm 1 provides pseudocode for this procedure, and we prove that it converges linearly to the target level set in Appendix A.

Algorithm 1 HarnackTrace($\mathbf{r}_0, \mathbf{v}, f^*, f(\mathbf{x}), R(\mathbf{x}), c(\mathbf{x}), \varepsilon, t_{\max}$)

Input: A ray origin $\mathbf{r}_0 \in \mathbb{R}^3$, unit ray direction $\mathbf{v} \in \mathbb{R}^3$, target level set value $f^* \in \mathbb{R}$, harmonic function $f : \mathbb{R}^3 \to \mathbb{R}$, a radius function $R : \mathbb{R}^3 \to \mathbb{R}_{>0}$, lower bound function $c : \mathbb{R}^3 \to \mathbb{R}$, stopping tolerance $\varepsilon > 0$, and maximum ray time $t_{\max} > 0$.

Output: The time t^* of the first intersection, or -1 if no intersection occurs.

```
1:  t ← 0
2:  do
3:      r_t ← r_0 + t v                          ▷ current point along ray
4:      f_t ← f(r_t)                             ▷ current function value
5:      if |f_t − f*| ≤ ε‖∇f(r_t)‖ then           ▷ stopping condition (§3.1.2)
6:          return t
7:      R_t ← R(r_t)                             ▷ radius of ball used to bound step size
8:      c_t ← c(r_t)                             ▷ shift that makes f positive on ball B_{R_t}(r_t)
9:      if f* ≤ c_t then                         ▷ if f* lies below the lower bound...
10:         ρ ← R                                ▷ ...we can safely take the maximum step of R
11:     else
12:         a ← (f_t − c_t)/(f* − c_t)            ▷ otherwise, shift f and...
13:         ρ ← ½ R_t |a + 2 − √(a² + 8a)|       ▷ ...compute a safe step size
14:     t ← t + ρ                                ▷ take step
15: while t < t_max
16: return −1                                    ▷ ray does not hit surface
```

3.1.1 Radii and Bounds. The only challenge in applying Harnack tracing to a given application scenario is determining values for R and c. An important observation, following from lines 12 and 13 of Algorithm 1, is that the largest step size will be achieved by using (i) the tightest possible lower bound c on f and (ii) the largest possible radius R. The reason is that the step size ρ approaches the maximum step size $\rho = R$ as $c \to f(\mathbf{r}_t)$, and approaches zero as $c \to -\infty$ (Figure 3). Simultaneously, the step size grows in proportion to the ball radius R. However, for harmonic functions (which are saddle-like everywhere) larger balls inevitably contain more negative values. To achieve efficient computation, one must hence balance the choice of R and c.

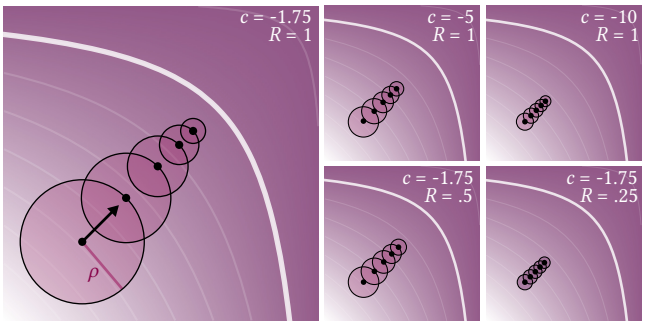


Fig. 3. To determine a conservative step size ρ, we need a lower bound c on the value of the harmonic function within a ball of radius R. As seen here, tighter bounds yield larger step sizes, as do larger radii. However, these two goals are often in conflict, as larger balls will contain lower values—obtaining efficient step sizes requires balancing these two considerations.

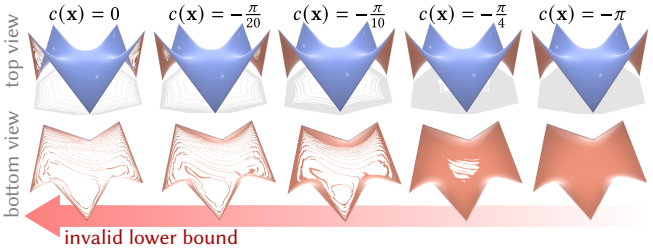


Fig. 4. Harnack tracing requires a lower bound $c(\mathbf{x})$ on the implicit function $f(\mathbf{x})$. If c is invalid—*i.e.* fails to bound f—then rays may pass through the surface, yielding artifacts which become more severe as c increases *(left)*. The artifacts are also worse for rays approaching from below ($f < f^*$), rather than above ($f > f^*$), as f takes on higher values above the surface. We describe how to obtain valid bounds in a variety of important cases.

Lipschitz vs. Harnack Bounds. This situation is not so different from classic sphere tracing: to render any new class of implicit functions, one must derive problem-specific bounds—as done gradually over the years for Lipschitz sphere tracing (Section 5.2.2). In both cases, however, finding *some* reasonable bound is typically possible, even if the *optimal* bound is hard to find. We walk through the derivation of several lower bounds in Section 4, enabling us to apply Harnack tracing in a variety of application scenarios. As in the Lipschitz case, one can also (as a fallback) simply "guess and check", adjusting c until the algorithm produces a valid result (Figure 4). Moreover, one can always take the maximum step size among any collection of conservative bounds (since all are "safe")—including both Lipschitz *and* Harnack bounds. In this sense, the Harnack approach can be viewed as complementary to the Lipschitz approach, rather than as a "competitor."

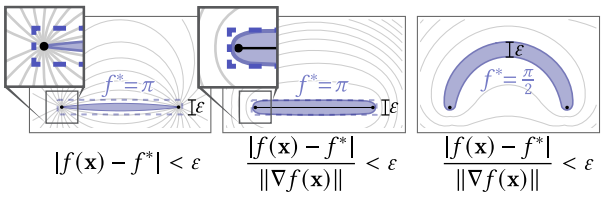


Fig. 5. If $f(\mathbf{x})$ is a signed distance function, then terminating intersection queries when $|f(\mathbf{x}) - f^*| < \varepsilon$ ensures that $\mathbf{x}$ is within ε of the chosen level set. But, when $f(\mathbf{x})$ is a general function, this condition loses its geometric meaning and produces an uneven profile along the target surface *(left)*. We can obtain a more meaningful stopping condition using the gradient $\nabla f(\mathbf{x})$, to relate changes in function value to changes in position *(center, right)*.

3.1.2 Stopping Condition. When f is a signed distance function, a natural stopping condition is to ask that $|f(\mathbf{x}) - f^*| < \varepsilon$, which ensures that $\mathbf{x}$ lies within a small geometric distance ε of the level set f^*. When f is harmonic, however, this stopping condition does not carry the same geometric meaning: $f(\mathbf{x})$ can be very far from—or unnecessarily close to—the target surface. To get a more meaningful stopping criterion near the surface, we hence use the condition

$$\frac{|f(\mathbf{x}) - f^*|}{\|\nabla f(\mathbf{x})\|} < \varepsilon,$$

which gives a good estimate of distance for values of $f(\mathbf{x})$ near f^* provided that f is sufficiently smooth. As seen in Figure 5, this condition ensures that rays terminate in a region of near-uniform thickness, whereas the naïve condition $|f(\mathbf{x}) - f^*| < \varepsilon$ leads to significant variability. This uniformity is especially important near singularities, where f has a steep gradient and requires extremely small steps to meet the stringent requirements of the naïve condition—Figure 22 visualizes the iteration counts resulting from both stopping conditions. Uniformity is also essential when shooting secondary rays (*e.g.* to render shadows), where error estimates for the point of intersection are needed to ensure that the outgoing ray does not get "stuck" at the point of departure [Pharr et al. 2023, Chapter 6.8].

3.1.3 Surface Normals. We can compute a unit normal $\mathbf{n}$ at any intersection point $\mathbf{x}$ by simply normalizing the gradient of our function f, *i.e.*, $\mathbf{n} = \nabla f / \|\nabla f\|$. This gradient can be computed using a closed-form or automatically-derived expression if available, or any standard finite difference approximation. For finite differences, we use the tetrahedral scheme advocated by Quilez [2015]; however, even careful treatment of finite differences can yield visual artifacts on singular functions (see *e.g.* Figure 12). We hence also give closed-form gradient expressions for specific examples in Section 4.

3.1.4 Acceleration. The "over stepping" technique used by Keinert et al. [2014, §3.1] for sphere tracing also provides a simple and effective method to accelerate Harnack tracing. The core insight is that at iteration k, a step is safe so long as it is less than the *sum* of the safe step sizes ρ_k from iteration k and ρ_{k+1} from iteration $k+1$: we know that the target level set cannot come within ρ_k of our position at iteration k *or* within ρ_{k+1} of our position at iteration $k+1$. So rather than taking steps of size ρ_k, we can use a larger step size δt, and then check whether $\delta t \leq \rho_k + \rho_{k+1}$. In practice, we set $\delta t \leftarrow 1.75\rho_k$, falling back to a step of size ρ_k if δt is too large.

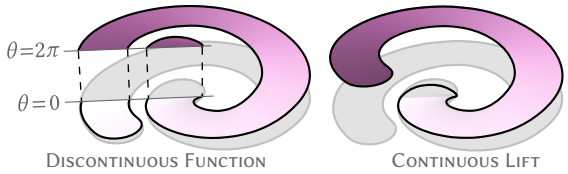


Fig. 6. Many harmonic functions naturally arise as angle-valued functions. Although such functions appear discontinuous when plotted in the range $[0, 2\pi)$, they can always be lifted to form a continuous harmonic function on any simply-connected domain. Importantly, we never need to construct this lift explicitly: its mere existence is sufficient for the bound to hold.

3.2 Angle-Valued Functions

An important class of harmonic functions is those that are angle-valued (see especially Sections 4.2–4.6). More precisely, we say a real-valued function is *angle-valued* if it is continuous modulo 2π, *i.e.*, real values that differ by an integer multiple of 2π encode the same angle. In this way, functions with discontinuous real values can still describe continuously-varying angles (*e.g.*, Section 1, *inset*).

Although angle-valued functions are often computed using expressions that give values in the range $[0, 2\pi)$ or $[-\pi, \pi)$, yielding apparent discontinuities, they can locally be lifted to a continuous function (Figure 6). Importantly, *we never need to explicitly compute or construct such a lift*: its mere existence is sufficient for the Harnack inequality to be valid. The only challenge is determining the range of the lifted function, in order to apply the inequality. In particular, if we know that a function $f : \Omega \to [0, 2\pi)$ jumps discontinuously at most k times up or k times down along any segment within Ω emanating from $\mathbf{x}_0$, then we immediately have a lower bound of $-2\pi k$. Hence, if we can bound the number of signed intersections with the target surface (*i.e.* the number of jumps up minus the number of jumps down), we can obtain a lower bound c on the lifted function, and can still use Harnack tracing to find the *first* intersection.

Algorithm 2 gives a modified procedure suitable for angle-valued functions. To make this algorithm applicable to a wider variety of use cases, we allow the angle-valued function $f(\mathbf{x})$ to have period $2\pi\omega$ for some constant ω, rather than always having period 2π. This generality allows us to handle, *e.g.*, the solid angle function in Section 4.2 with period 4π. A small but important change relative to Algorithm 1 is that we must bound the step size according to both the closest level set values f_+ and f_- above and below the current value f (*resp.*), and must return a hit if we come sufficiently close to either level set. Otherwise, the algorithm is the same.

3.3 Height Fields

Given a continuous harmonic function $f : \mathbb{R}^2 \to \mathbb{R}$ on the plane, we can use Harnack tracing to intersect a ray with the height field

$$\mathcal{S} := \{(x_1, x_2, x_3) \mid f(x_1, x_2) = x_3\}.$$

Since $\mathcal{S}$ is the zero level set of $\phi(x_1, x_2, x_3) := f(x_1, x_2) - x_3$, which is harmonic on $\mathbb{R}^3$ whenever f is harmonic on $\mathbb{R}^2$, we can find intersections with $\mathcal{S}$ by calling Algorithm 1 on ϕ. We can also compute intersections with height fields of angle-valued functions by treating ϕ as an angle-valued function on $\mathbb{R}^3$ (Section 4.6).

Algorithm 2 TraceAngleValued($\mathbf{r}_0, \mathbf{v}, \omega, \phi, f(\mathbf{x}), R(\mathbf{x}), c(\mathbf{x}), \varepsilon, t_{\max}$)

Input: A ray origin $\mathbf{r}_0 \in \mathbb{R}^3$, unit ray direction $\mathbf{v} \in \mathbb{R}^3$, an angular frequency $\omega \in \mathbb{R}_{>0}$ and phase shift $\phi \in \mathbb{R}$ that determine target level set values $f^*(k) := 2k\pi\omega + \phi$ for all $k \in \mathbb{Z}$, harmonic function $f : \mathbb{R}^3 \to \mathbb{R}$, a radius function $R : \mathbb{R}^3 \to \mathbb{R}_{>0}$, lower bound function $c : \mathbb{R}^3 \to \mathbb{R}$, stopping tolerance $\varepsilon > 0$, and maximum ray time $t_{\max} > 0$.

Output: The time t^* of the first intersection with any level set $f^*(k)$, or -1 if no intersection occurs.

$t \leftarrow 0$
do
  $\mathbf{r}_t \leftarrow \mathbf{r}_0 + t\mathbf{v}$ ▷ *current point along ray*
  ▷ *Find the two level set values bracketing the current value of f*
  $f_0 \leftarrow (f(\mathbf{r}_t) - \phi)/(2\pi\omega)$
  $f_- \leftarrow (2\pi\omega)\lfloor f_0 \rfloor + \phi$
  $f_+ \leftarrow (2\pi\omega)\lceil f_0 \rceil + \phi$
  ▷ *Stop if close to either surface (§3.1.2)*
  if $\min(f(\mathbf{r}_t) - f_-, f_+ - f(\mathbf{r}_t)) \leq \varepsilon\|\nabla f(\mathbf{r}_t)\|$ **then**
    return t
  $R_t \leftarrow R(\mathbf{r}_t)$ ▷ *radius of ball used to bound step size*
  $c_t \leftarrow c(\mathbf{r}_t)$ ▷ *shift that makes f positive on ball* $B_{R_t}(\mathbf{r}_t)$
  ▷ *Compute a step size bound for each of the two closest level sets*
  $a_- \leftarrow (f(\mathbf{r}_t) - c_t)/(f_- - c_t)$
  $a_+ \leftarrow (f(\mathbf{r}_t) - c_t)/(f_+ - c_t)$
  $\rho_- \leftarrow \frac{1}{2}R_t \left| a_- + 2 - \sqrt{a_-^2 + 8a_-} \right|$
  $\rho_+ \leftarrow \frac{1}{2}R_t \left| a_+ + 2 - \sqrt{a_+^2 + 8a_+} \right|$
  ▷ *Take the smaller of the two steps*
  $\rho \leftarrow \min(\rho_-, \rho_+)$
  $t \leftarrow t + \rho$
while $t < t_{\max}$
return -1 ▷ *ray does not hit surface*

4 EXAMPLES AND EVALUATION

Harmonic functions and their level sets play an important role across geometric and visual computing—here we explore how Harnack tracing can be applied to several example use cases. We start with harmonic polynomials as a didactic example (Section 4.1), before introducing a novel strategy for visualizing nonplanar polygons (Section 4.2). We also consider surface reconstruction (Section 4.3), mesh repair (Section 4.4), architectural design (Section 4.5), and mathematical visualization (Section 4.6). As a final example, we use Harnack tracing to visualize a function which is *not* harmonic (Section 4.7), paving the way to potentially broader applications (Section 4.7.1). We then conclude with a discussion of our implementation (Section 4.8), how many iterations it takes to find intersections (Section 4.9) and its convergence rate (Section 4.10).

4.1 Spherical Harmonics

As a simple example, which does not involve any discontinuities, we start by visualizing *spherical harmonics* as level sets of harmonic polynomials on $\mathbb{R}^3$ (Figure 7). Though often defined using Legendre functions in polar coordinates, any spherical harmonic can also be expressed as the restriction of a homogeneous harmonic polynomial

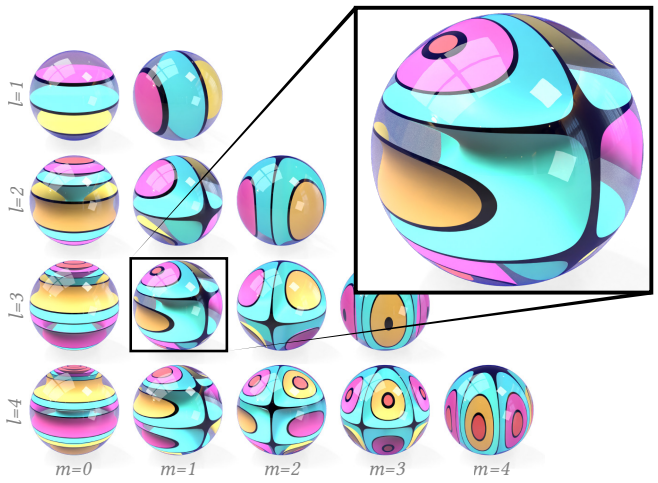


Fig. 7. Harmonic polynomials provide an elementary example of harmonic functions. When restricted to the sphere, these polynomials describe the *spherical harmonics*. Here we visualize each spherical harmonic by restricting the level sets of the associated spherical polynomial to the unit ball.

$p(x_1, x_2, x_3)$ to the unit sphere $S^2 := \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\| = 1\}$. Curves where the level sets of p meet the sphere trace out contours of the spherical harmonic.

Explicitly, a polynomial $p : \mathbb{R}^3 \to \mathbb{R}$ is harmonic if it satisfies Laplace's equation (Equation 1). Take for instance the polynomial

$$p(x_1, x_2, x_3) := x_1^2 x_2 - x_2 x_3^2, \tag{8}$$

for which $\Delta p = 2x_2 - 2x_2 = 0$. To visualize a level set of p within the unit ball we start Harnack tracing at the time t_0 where our ray first pierces the unit sphere S^2 and set $t_{\max}$ to the time where the ray exits the sphere. At each point $\mathbf{x}$ within this sphere, we then need a radius R and value c such that the shifted harmonic polynomial $\tilde{p}(\mathbf{x}) := p(\mathbf{x}) - c$ is positive over the ball $B_R(\mathbf{x})$. For the radius, one might try the distance to the unit sphere $1 - \|\mathbf{x}\|$, but this choice yields an invalid value of $R = 0$ on the unit sphere itself—making it impossible to start Harnack tracing from a point on the unit sphere. Instead, we let R be the distance to a slightly larger sphere, of radius $h > 1$, yielding positive values of R at all points inside the unit sphere. In this case, our function is particularly simple, and we can obtain a good lower bound c by analytically minimizing $p(x)$ over the ball $B_h(0)$. For instance, for the polynomial in Equation 8 we have $c(h) = -(2h^3)/(3\sqrt{3})$. This scenario illustrates the need to balance c and R: if we make h much bigger, to increase the radii R (and hence our step sizes), the lower bound $c(h)$ rapidly becomes more negative—eliminating any advantage of a larger R. In our examples, we therefore use a value $h = 1.25$, only slightly larger than the unit sphere. Concretely, this amounts to running Algorithm 1 with the function $f(\mathbf{x})$ given by Equation 8, radius function $R(\mathbf{x}) := 1.25 - \|\mathbf{x}\|$, and the constant bound $c(\mathbf{x}) := -(2 \cdot 1.25^3)/(3\sqrt{3}) \approx -0.75$.

Gradient evaluation. Polynomials are continuous and smooth, so we evaluate their gradients via finite differences (Section 3.1.3.)

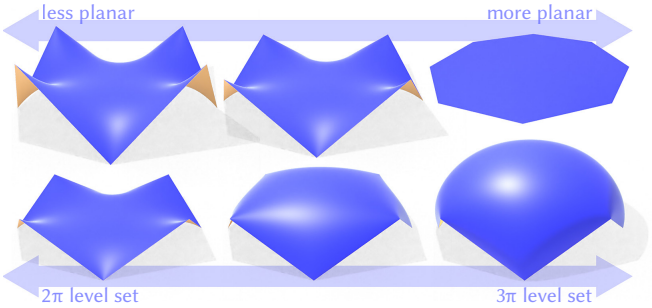
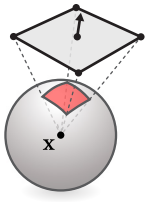


Fig. 8. We define the geometry of a nonplanar polygon to be a level set of its *solid angle* function, which is harmonic. This definition provides well-behaved geometry even when the boundary is highly nonplanar, and varies smoothly as the boundary changes *(top)*. By taking different level sets, one can adjust the convexity/concavity of the interpolating geometry *(bottom)*.

4.2 Nonplanar Polygons

A perennial question in computer graphics is how to interpret polygons whose vertices do not sit in a common plane [Bunge et al. 2023]. The ability to ray trace harmonic functions provides an elegant answer: we can visualize a nonplanar polygon as a level set of a harmonic function naturally associated to the polygon, namely, its *(signed) solid angle* [Binysh and Alexander 2018]. This definition, illustrated in Figure 8, has many attractive properties relative to existing nonplanar interpolation schemes. For quadrilaterals, for instance, simple bilinear interpolation can yield severe foldover (Figure 9). For more general nonplanar n-gons, one observes similar foldover with mean value coordinates [Floater 2003], harmonic coordinates [Joshi et al. 2007], Catmull-Clark subdivision surfaces[1] [Catmull and Clark 1978], and the virtual vertex scheme of Bunge et al. [2020] (Figure 10). Beyond these comparisons, using the signed solid angle allows us to visualize far more general nonplanar polygons that can have holes, or even knotted/linked boundaries (Figure 13). Classic *minimal surfaces* can also interpolate at this level of generality, but must be computed using an explicit mesh [Pinkall and Polthier 1993] or dense computational grid [Wang and Chern 2021]. An especially beautiful feature of the harmonic definition is that, like Wang and Chern, we need not explicitly prescribe the topology of our interpolating surface *a priori* (Figure 11).

4.2.1 Solid angle. Intuitively, solid angle is the size of the "shadow" cast by a surface patch onto a unit sphere around the evaluation point $\mathbf{x}$ by central projection. The *signed* solid angle takes relative orientation into account, changing sign if the normal of the surface patch is reversed. Unlike physical shadows, it also accounts for multiple covering: the area of each overlapping piece adds to (or subtracts from) the total solid angle.



More precisely, consider a smooth surface patch $\Sigma \subset \mathbb{R}^3$. The signed solid angle of Σ at a point $\mathbf{x}$ is given by the integral

$$\Omega_\Sigma(\mathbf{x}) := \int_\Sigma \frac{\mathbf{n}(\mathbf{p}) \cdot (\mathbf{p} - \mathbf{x})}{\|\mathbf{p} - \mathbf{x}\|^3} d\mathbf{p}, \tag{9}$$

[1]with boundary preservation, as implemented in Blender [2023]

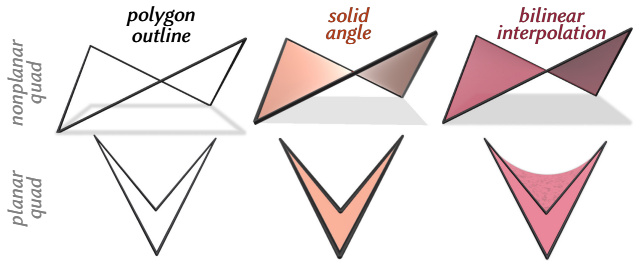


Fig. 9. A natural way to interpolate quadrilaterals is with a bilinear patch—however, such patches can exhibit foldover, even for planar quads. Our Harnack-based interpolation scheme exhibits no such foldover.

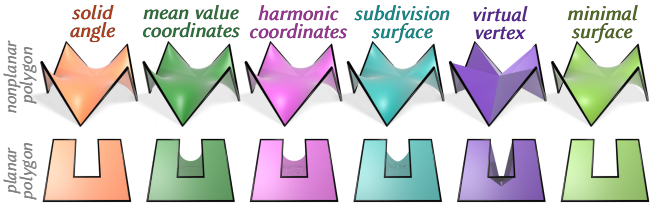


Fig. 10. Given the boundary of a polygon, how should one fill its interior? Here we compare our harmonic definition with several standard schemes (for mean value and harmonic coordinates, we interpolate vertex positions over a regular n-gon). Apart from minimal surfaces—which require a fine mesh to compute—our definition is the only one that avoids foldover.

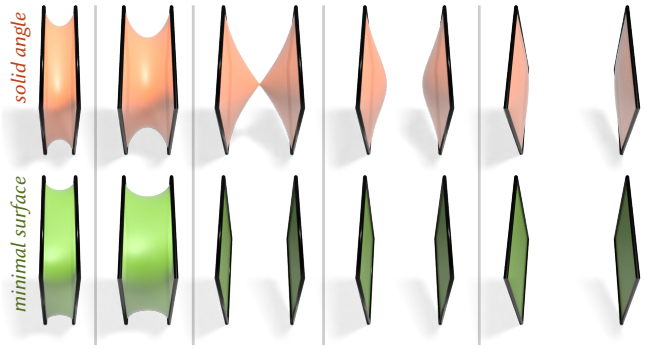


Fig. 11. For many curves, our nonplanar polygon definition looks quite similar to a minimal surface. However, they differ in a key way: minimal surfaces can jump discontinuously as the boundary curves vary *(bottom)*, whereas our harmonic level sets always vary continuously *(top)*.

where $\mathbf{n}(\mathbf{p})$ is the unit normal to Σ at $\mathbf{p}$. Remarkably, taking the value of this integral modulo 4π yields a continuous function which depends only on the boundary of the surface patch, independent of its interior [Binysh and Alexander 2018, §1]. So, we can define a canonical solid angle function $\Omega_P(\mathbf{x})$ for a possibly-nonplanar polygon P by taking the solid angle of any surface Σ spanning P.

Most importantly for our purposes, this solid angle function is well-known to be a harmonic angle-valued function with period 4π [Binysh and Alexander 2018, §1]. More precisely, it is harmonic away from the edges of P itself, and exhibits singular behavior in

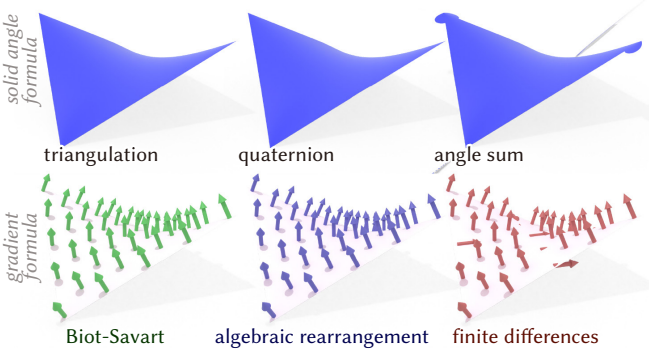


Fig. 12. Not all expressions for the solid angle or its derivative provide accurate results in floating point. *Top*: the solid angle formulas based on triangulation and quaternions work well, but the expression based on angle sums suffers from numerical instability. *Bottom*: The Biot-Savart law and its rearrangement by Adiels et al. [2022] both yield accurate normals, but finite differences give incorrect results due to jumps in the angle-valued function.

the vicinity of these edges—where the angle quickly goes through a full period of 4π. Since the solid angle takes every possible value as we go around any point of the boundary curve, every level set of the solid angle function will provide a surface spanning the curve. In the special case where P is planar, the 2π level set yields the planar region bounded by the curve. Hence, even in the nonplanar case, we use the 2π level set to define the surface bounded by P.

4.2.2 Numerics. There are many expressions for the signed solid angle of a polygon, which are all equivalent (modulo 4π) in exact arithmetic. In floating point, however, not all expressions work equally well when ray tracing (Figure 12). We considered three expressions for solid angle: (i) direct calculation via a triangulation, (ii) the quaternionic scheme of Chern and Ishida [2023, Cor. 3.4.1], and (iii) the angle sum formula of Legendre [1817, §505], as well as three expressions for its gradient: (i) a direct expression via the Biot-Savart formula, (ii) a rearrangement of this expression suggested by Adiels et al. [2022, Eq. 10], and (iii) a finite difference approximation. Our preferred expressions, described below, are the triangulation method for solid angle and the Biot-Savart formula for its gradient. Details on the other formulations can be found in the supplement.

Function evaluation. We denote the vertices of the polygon P by $\mathbf{p}_1, \ldots, \mathbf{p}_k \in \mathbb{R}^3$. To evaluate its solid angle, we triangulate P and sum the solid angles of each triangle. For symmetry, we triangulate P by connecting each vertex to a point $\mathbf{z} \in \mathbb{R}^3$ at the average of the vertex positions. To evaluate the solid angle of a triangle, we use the formula of van Oosterom and Strackee [1983]. In particular, letting

$$\mathbf{a} \coloneqq \mathbf{p}_i - \mathbf{x}, \quad \mathbf{b} \coloneqq \mathbf{p}_{i+1} - \mathbf{x}, \quad \mathbf{c} \coloneqq \mathbf{z} - \mathbf{x},$$

and letting a, b, c be the magnitudes of $\mathbf{a}, \mathbf{b}, \mathbf{c}$, *resp.*, the signed solid angle of triangle $\mathbf{p}_i\mathbf{p}_{i+1}\mathbf{z}$ is given by

$$\Omega_{\text{tri}}(\mathbf{x}) \coloneqq 2\,\text{atan2}\Big(\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c}), abc + (\mathbf{a}\cdot\mathbf{b})c + (\mathbf{b}\cdot\mathbf{c})a + (\mathbf{a}\cdot\mathbf{c})b\Big). \quad (10)$$

It is important to use the two-argument arc tangent function $\text{atan2}(y, x)$, yielding values in the range $[-\pi, \pi)$, rather than the range $[-\pi/2, \pi/2]$ of the ordinary arc tangent function. For efficiency, one can also reduce the number of atan2 evaluations by applying the identity

$$\text{atan2}(a, b) + \text{atan2}(c, d) \equiv \text{atan2}(ad + bc, bd - ac) \pmod{2\pi}.$$

Gradient evaluation. We evaluate the gradient of solid angle as

$$\nabla\Omega_P(\mathbf{x}) = \sum_{i=1}^{k} (\mathbf{g}_i - \mathbf{g}_{i+1}) \cdot \left(\frac{\mathbf{g}_i}{\|\mathbf{g}_i\|} - \frac{\mathbf{g}_{i+1}}{\|\mathbf{g}_{i+1}\|}\right) \frac{\mathbf{g}_i \times \mathbf{g}_{i+1}}{\|\mathbf{g}_i \times \mathbf{g}_{i+1}\|^2}, \quad (11)$$

where $\mathbf{g}_i \coloneqq \mathbf{p}_i - \mathbf{x}$. This formula can be derived by integrating the Biot-Savart formula along each line segment in the polygon's boundary (see *e.g.* Equation 8 of Adiels et al. [2022]).

Function bounds and bounding boxes. When using Harnack tracing on level sets of $\Omega_P(\mathbf{x})$, we take $R(\mathbf{x})$ to be the distance from $\mathbf{x}$ to the polygonal curve P and set $c(\mathbf{x}) = -4\pi$. In Appendix C, we show that this bound is valid when the curve is connected, intersection-free and lies on the boundary of a convex domain. Although it is not guaranteed to apply to all polygons, we have not found an example where Harnack tracing using this bound fails, even for polygons with holes or knotted boundaries like those depicted in Figure 13.

In order to incorporate Harnack tracing into an existing renderer, it is also helpful to have bounds on the spatial extent of the target level set. In Appendix B, we show that under the same assumptions on P, the 2π level set of $\Omega_P(\mathbf{x})$ is contained within the convex hull of P, and hence within any bounding box of P.

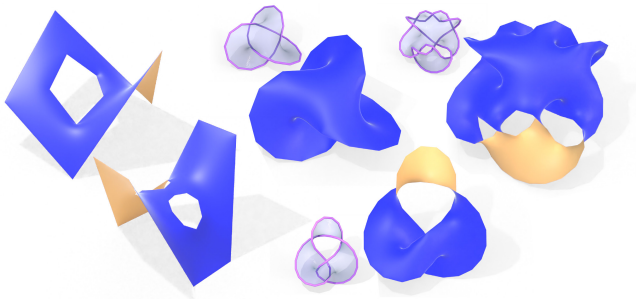


Fig. 13. Our nonplanar polygon definition automatically applies to difficult cases like polygons with holes *(left)*, or knotted boundaries *(right)*.

4.2.3 Polygons with complex topology. Some boundary representations used in computer-aided design (CAD) allow polygons to have holes. To define nonplanar polygons with holes, we can simply add the solid angles of all boundary components (assuming consistent orientation) to define a single harmonic function for a polygon with holes—Figure 13, *left* shows some examples. Moreover, while polygons arising from meshes are almost universally homotopic to the unknot, nothing about our formulation prevents us from rendering curves that are knotted, or collections of curves that are linked. As illustrated in Figure 13, *right*, the solid angle level sets associated to knotted polygons still form smooth surfaces bounded by the polygon, yielding so-called "Seifert surfaces." Unlike existing methods, which first build topologically-valid Seifert surfaces and then smooth them [van Wijk and Cohen 2006], we can directly visualize smooth interpolating surfaces, sidestepping the usual numerical challenges associated with mesh-based optimization.

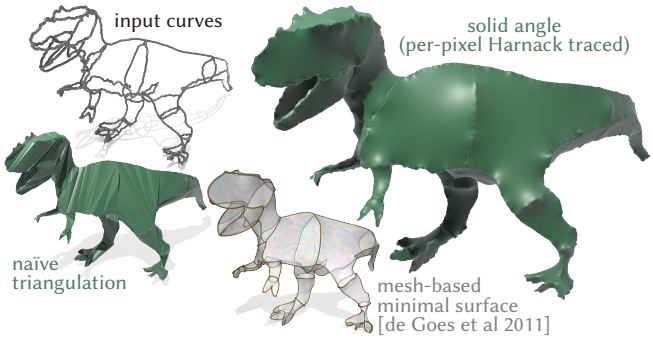


Fig. 14. Given a sparse "exoskeleton" approximating a surface *(top left)* we can directly ray trace an interpolating surface *(top right)*. The resulting image is both higher-quality than the simple triangulation used by most mesh viewers *(bottom left)*, and much simpler to compute than optimizing a mesh-based minimal surface, as originally proposed by de Goes et al. [2011].

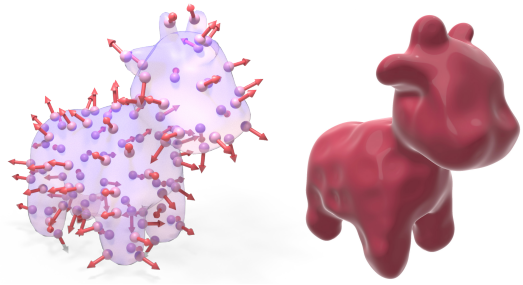


Fig. 15. Given an oriented point cloud *(left)*, we can directly visualize an interpolating surface *(right)*. This procedure effectively shows the result of running the Poisson surface reconstruction algorithm of Kazhdan et al. [2006], without requiring any volumetric meshing or linear solves.

4.2.4 Exoskeletons. An extreme case of nonplanar polygons, we consider the "exoskeleton" curve networks computed by de Goes et al. [2011] as concise descriptions of a surface. As depicted in Figure 14, we can treat the exoskeleton as a nonplanar polygon mesh of high degree and Harnack trace the resulting surface, avoiding the minimal surface computation originally proposed by de Goes et al.

4.3 Point Clouds (Poisson Reconstruction)

We can directly visualize a surface interpolating a given point cloud without generating an explicit surface representation (Figure 15). Effectively, we draw the surface that would be generated by the *Poisson surface reconstruction* algorithm of Kazhdan et al. [2006] (in the limit as the kernel size goes to zero).

Suppose we are given a collection of points $\mathbf{p}_1, \ldots, \mathbf{p}_k \in \mathbb{R}^3$ along with normal vectors $\mathbf{n}_1, \ldots, \mathbf{n}_k \in \mathbb{R}^3$, and their associated areas $a_1, \ldots, a_k \in \mathbb{R}_{>0}$ (*e.g.*, acquired via scanning or estimated *à la* Barill et al. [2018, §3.1]). As discussed by Barill et al. [2018, §2.1], one can reconstruct surfaces from these points by considering level sets of the *dipole potential*

$$f(\mathbf{x}) := \sum_{i=1}^{k} a_i \frac{(\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i}{\|\mathbf{p}_i - \mathbf{x}\|^3}, \tag{12}$$

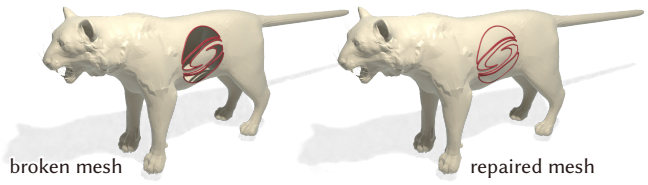


Fig. 16. Given a surface mesh with imperfections such as holes *(left)*, we can directly visualize the repaired surface defined via the *generalized winding number (right)*, reproducing the example from Jacobson et al. [2013, Figure 1].

which is a harmonic function with singularities at the points $\mathbf{p}_i$. We can then ray trace level sets of this potential using Harnack tracing.

Here we must be careful due to the singular nature of the dipole potential. For one thing, our gradient termination condition yields artifacts near points $\mathbf{p}_i$, since here f is not well-approximated by a linear function (see inset). Instead, we terminate rays if either (i) the value of f is within a small tolerance $\varepsilon > 0$ of zero, or (ii) the ray is closer than a small distance $d > 0$ of any point $\mathbf{p}_i$. Also, to ensure that $f(\mathbf{x})$ remains bounded on the ball $B_R(\mathbf{x})$, we take the radius $R(\mathbf{x})$ to be a fraction $\alpha \in (0, 1)$ of the distance to the closest point:

$$R(\mathbf{x}) := \alpha \min_i \|\mathbf{p}_i - \mathbf{x}\|. \tag{13}$$

In our examples we found $\alpha = 1/4$ helped minimize the number of steps. By considering the numerator and denominator of each term in $f(\mathbf{x})$ separately, we can then obtain a bound on its value over any ball $B_R(\mathbf{x})$ that does not contain any of the points $\mathbf{p}_i$:

$$c(\mathbf{x}) := \sum_{i=1}^{k} \frac{a_i \big((\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R(\mathbf{x}) \big)}{\big(\|\mathbf{p}_i - \mathbf{x}\| + \operatorname{sign}\big((\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R(\mathbf{x}) \big) R(\mathbf{x}) \big)^3}. \tag{14}$$

A more detailed derivation can be found in Appendix D. Using the definitions of f, R, and c from this section, we now have everything needed to execute Algorithm 1.

Gradient evaluation. To shade the surface, we use the following closed-form expression for the gradient of $f(\mathbf{x})$:

$$\nabla f(\mathbf{x}) = \sum_{i=1}^{k} a_i \left(3 \frac{(\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i}{\|\mathbf{p}_i - \mathbf{x}\|^5} (\mathbf{p}_i - \mathbf{x}) - \frac{1}{\|\mathbf{p}_i - \mathbf{x}\|^3} \mathbf{n}_i \right). \tag{15}$$

If a ray terminates due to hitting one of the points $\mathbf{p}_i$, we take the corresponding normal $\mathbf{n}_i$ as the surface normal.

4.4 Mesh Repair (Generalized Winding Number)

We can also directly visualize a surface that "fills in" and/or fixes defects in a polygon mesh with cracks and self-intersections. Effectively, we directly visualize the results of the *generalized winding number* scheme of Jacobson et al. [2013], again without explicitly meshing the surface (Figure 16). This is possible because the generalized winding number defined by Jacobson et al. is precisely the signed solid angle of the mesh faces, so the techniques of Section 4.2 can be applied without requiring any changes.

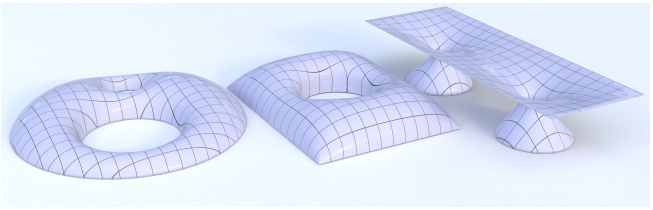


Fig. 17. Here we use Harnack tracing to directly visualize a special class of *grid shells* used in architecture, reproducing examples from Adiels et al. [2022] figures 11, 12, and 22 (*resp.*). To do so, we extend our algorithm to handle circular holes in addition to polygons.

4.5 Architectural Grid Shells

Adiels et al. [2022] make a beautiful observation connecting level sets of the solid angle function for a curve (*à la* Section 4.2) to shell structures for architectural geometry—namely, if the boundary curve is planar, then these surfaces will automatically have principal curvature networks aligned to the boundary curve (needed for panelization) and will exhibit an approximately constant span-to-height ratio (making them structurally viable as grid shells). The fact that surfaces with such nice geometric and structural properties arise from a simple function like solid angle is quite remarkable: ordinarily one must perform explicit meshing and optimization to get surfaces suitable for architecture. However, Adiels et al. still do not take full advantage of this elegant representation when exploring the design space, since they must still build a mesh for visualization purposes. (In particular, they sample random points and use Newton's method to push them onto a given level set.) We can instead visualize these surfaces directly via Harnack tracing (Figure 17).

To do so, we generalize the algorithm for nonplanar polygons to also compute the solid angles of circles. Using the notation of Figure 18, the signed solid angle subtended by a circle is given by

$$\Omega(\mathbf{x}) = \begin{cases} 2\pi - \frac{2L}{R_{\max}} K(k^2) + \frac{2L}{R_{\max}} \frac{r_0 - r_m}{r_0 + r_m} \Pi(\alpha^2, k^2) & r_0 < r_m \\ \pi - \frac{2L}{R_{\max}} K(k^2) & r_0 = r_m \\ -\frac{2L}{R_{\max}} K(k^2) + \frac{2L}{R_{\max}} \frac{r_0 - r_m}{r_0 + r_m} \Pi(\alpha^2, k^2) & r_0 > r_m, \end{cases} \tag{16}$$

where $\alpha^2 := (4 r_0 r_m)/(r_0 + r_m)^2$ and $k^2 := 1 - R_1^2/R_{\max}^2$ (see Paxton [1959] or Rothe [1969] for derivations). $K(m)$ and $\Pi(\alpha^2, m)$ are complete elliptic integrals of the first and third kinds, which can be evaluated using standard numerical packages [Galassi et al. 2009]. The rest of the algorithm remains the same as Section 4.2; in particular, we take $R(\mathbf{x})$ to be the distance to the curve and $c(\mathbf{x}) = -4\pi$.

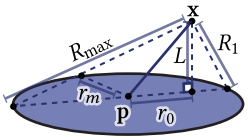


Fig. 18. Quantities used to evaluate $\Omega(\mathbf{x})$. Here $R_1 = \sqrt{L^2 + (r_0 - r_m)^2}$ and $R_{\max} = \sqrt{L^2 + (r_0 + r_m)^2}$.

Gradient evaluation. Smythe [1989, §7.10] gives a formula for $\nabla\Omega$:

$$\begin{aligned} \nabla\Omega(\mathbf{x}) = &- \frac{2L}{r_0 R_{\max}} \left(-K(k^2) + \frac{r_m^2 + r_0^2 + L^2}{R_1^2} E(k^2) \right) \hat{r} \\ &- \frac{2}{R_{\max}} \left(K(k^2) + \frac{r_m^2 - r_0^2 - L^2}{R_1^2} E(k^2) \right) \hat{z}, \end{aligned} \tag{17}$$

where $E(m)$ denotes a complete elliptic integral of the second kind.

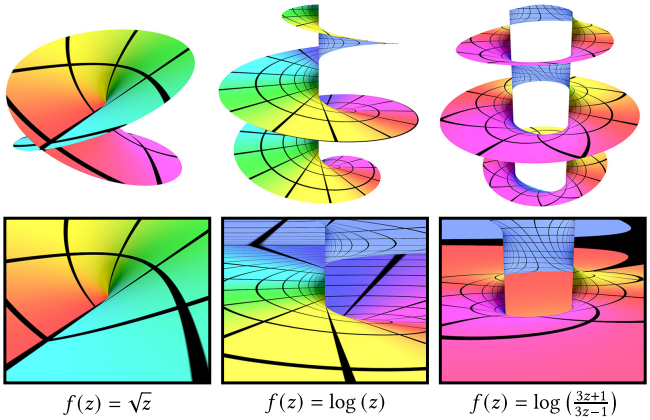
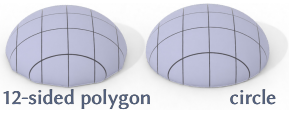


Fig. 19. Riemann surfaces are central objects of study in complex analysis. We can use Harnack tracing to render the surfaces associated to several standard complex functions, showing both the intersection with the unit ball *(top)*, and a full camera view of the surface *(bottom)*.

In practice, elliptic integrals can be expensive to compute, but can still be cheaper than applying the polygonal solid angle formula to a finely-meshed boundary curve. We find that using Equation 16 for the solid angle of a circle is about as expensive as evaluating the solid angle of a 12-sided polygon, while providing a smoother boundary curve (inset). If we use a 50-sided polygon, evaluating the polygon's solid angle takes twice as long as evaluating Equation 16.



4.6 Riemann Surfaces

Riemann surfaces associated to complex-differentiable functions like $\log(z)$ play a key role in complex analysis. Much of their interesting behavior occurs at singular points, which pose problems for standard techniques (*e.g.*, requiring extremely high resolution for marching cubes), but are accurately resolved by Harnack tracing (Figure 19).

Concretely, the imaginary part of a complex-differentiable function $f(z)$ is a real-valued harmonic function on $\mathbb{R}^2$. We can visualize the Riemann surface associated to f as the height field of $\operatorname{Im} f$. If f is multivalued, then $\operatorname{Im} f$ is a multivalued harmonic function. For instance, $\operatorname{Im}[\log(z)]$ is defined modulo shifts by 2π, making it angle-valued. So we can visualize the Riemann surface of $\log(z)$ by running TraceAngleValued (Algorithm 2) on the angle-valued function $\phi(x_0, x_1, x_2) := \operatorname{Im}[\log(x_0 + x_1 i)] - x_2$. For functions like $\sqrt{z}$, with a sign ambiguity rather than a shift, we modify lines 5–7 of Algorithm 2 to find the level set values bracketing the current value of $\phi(x_0, x_1, x_2) := \operatorname{Im}[f(x_0 + x_1 i)] - x_2$ by enumerating the possible values of $f(z)$. To render Figure 19 using Algorithm 2, we set $R(\mathbf{x})$ to the distance from $\mathbf{x}$ to the closest singularity and $c(\mathbf{x}) = -4$.

Gradient evaluation. The gradient of ϕ may be expressed using the complex derivative of f: $\nabla\phi = \left(\operatorname{Im}\left[\frac{\partial}{\partial z} f\right], \operatorname{Re}\left[\frac{\partial}{\partial z} f\right], -1\right)$. When the value of $\frac{\partial}{\partial z} f$ depends on the branch of the function, *e.g.* for $f(z) = \sqrt{z}$, we search through all possible values of $f(z)$ to identify which branch our intersection lies on before evaluating $\frac{\partial}{\partial z} f$.

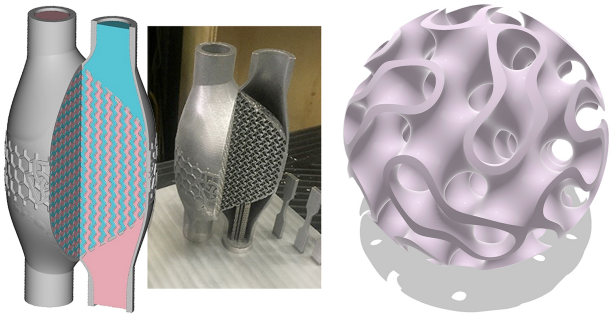
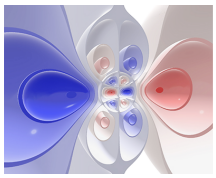


Fig. 20. The gyroid surface, commonly used in 3D manufacturing (*left*, reproduced from Diegel [2021]), is neither a signed distance function nor a harmonic function. However, it can easily be extended to a harmonic function in 4D. By ray tracing a "3D slice" of this function, we can directly visualize it via Harnack tracing *(right)*.

4.7 Beyond Harmonic Functions

A characteristic property of harmonic functions is that they exhibit no local minima or maxima—which would seem to be a significant limitation on the kinds of surfaces that can be visualized. However, one can apply Harnack tracing to harmonic functions in any higher dimension $\mathbb{R}^n$, and visualize a "slice" to obtain surfaces that cannot be realized as level sets of harmonic functions in 3D. An illustrative example is the ordinary sphere, which can be described by the $w = 0$ slice of the 4D function $f(x, y, z, w) = x^2 + y^2 + z^2 - 3w^2 - 1$. In general, one can try extending any implicit function $f(x, y, z)$ to a function $\bar{f}(x, y, z, w)$ such that (i) $\bar{f}$ is harmonic in $\mathbb{R}^4$, and (ii) $\bar{f}(x, y, z, 0) = f(x, y, z)$, *i.e.*, the restriction of $\bar{f}$ to the 3D subspace $w = 0$ yields the original function (and hence the desired surface).



For instance, as shown in the inset, we can visualize "hyperspherical" harmonics: 4D harmonic polynomials restricted to the 3-sphere and mapped into $\mathbb{R}^3$ via stereographic projection (see Appendix E.2 for details). A more interesting, non-polynomial example is the *gyroid*, described by the zero level set of the function

$$f_{\text{gyroid}}(x, y, z) = \sin(x)\cos(y) + \sin(y)\cos(z) + \sin(z)\cos(x). \quad (18)$$

The gyroid is widely used in 3D manufacturing due to, *e.g.*, its thermal and space-filling properties (Figure 20, *left*). The function $\bar{f}(x, y, z, w) \coloneqq e^{\sqrt{2}w} f_{\text{gyroid}}(x, y, z)$ provides a harmonic extension to $\mathbb{R}^4$—allowing us to Harnack trace the gyroid (Figure 20, *right*).

4.7.1 Arbitrary Implicit Functions. The gyroid example also suggests a strategy for Harnack tracing *arbitrary* implicit surfaces $f(x, y, z) = 0$, via frequency decomposition. The key observation is that Equation 18 is an example of a *Laplacian eigenfunction*, *i.e.*, a function ϕ such that $\Delta\phi = \lambda\phi$ for some eigenvalue $\lambda \in \mathbb{R}_{\leq 0}$. In particular, one can check that $\Delta f_{\text{gyroid}} = -2 f_{\text{gyroid}}$. More generally, for any such eigenfunction we have a harmonic extension

$$\bar{\phi}(x, y, z, w) \coloneqq e^{\sqrt{-\lambda}w}\phi(x, y, z), \quad (19)$$

whose Laplacian vanishes since

$$\begin{aligned}\Delta_{\mathbb{R}^4}\bar{\phi} &= \left(\Delta_{\mathbb{R}^4} e^{\sqrt{-\lambda}w}\right)\phi + \left(\nabla_{\mathbb{R}^4} e^{\sqrt{-\lambda}w}\right)\cdot\left(\nabla_{\mathbb{R}^4}\phi\right) + e^{\sqrt{-\lambda}w}\left(\Delta_{\mathbb{R}^4}\phi\right)\\ &= -\lambda e^{\sqrt{-\lambda}w}\phi + 0 + e^{\sqrt{-\lambda}w}\lambda\phi = 0.\end{aligned}$$

Hence, we could approximate any periodic implicit function $f : [0, 2\pi]^3 \to \mathbb{R}$ by a sum of Laplacian eigenfunctions, via the truncated Fourier transform

$$\hat{f}(\mathbf{x}) \coloneqq \sum_{k=0}^{N} \langle\langle f, \phi_k \rangle\rangle \phi_k(\mathbf{x}),$$

where $\langle\langle \cdot, \cdot \rangle\rangle$ denotes the L^2 inner product, and ϕ_k is the kth Laplacian eigenfunction. We can then harmonically extend $\hat{f}(\mathbf{x})$ to $\mathbb{R}^4$ by replacing each basis function ϕ_k with the corresponding function $\bar{\phi}_k$, *à la* Equation 19. Figure 21 shows the $\frac{1}{2}$ level set of the function

$$f(x, y, z) = \sin(6x)\sin(6y)\sin(6z) + \sin(2x)\sin(2y)\sin(2z),$$

which is neither harmonic nor a Laplacian eigenfunction.

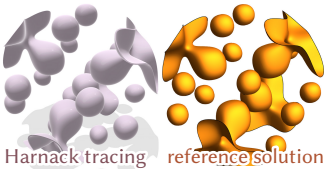


Fig. 21. Harnack tracing is not limited to saddle-like surfaces—in principle it can visualize any implicit function, via spectral expansion. Here: a sum of Laplacian eigenfunctions.

One can easily obtain a conservative bound c on such a sum by bounding each term individually and summing the bounds (see Appendix E for details). Finding a tight bound is more challenging since different terms in the sum can interfere constructively or destructively depending on their phase. The same problem arises when trying to find a tight Lipschitz bound on a sum of Fourier bases: for instance, even the simple functions $\sin(x)+\sin(2x)$ and $\sin(x)+\sin(2x+\pi/2)$ have different Lipschitz constants (3 and ∼2.7, *resp.*), despite having identical Lipschitz bounds for the individual terms. We hence consider this construction primarily to highlight the theoretical generality of Harnack tracing, and leave questions of efficiency to future work.

4.8 Implementation and Performance

We implemented our algorithm in two frameworks: as GLSL programs implemented via *ShaderToy* [Quilez and Jeremias 2013], to gauge efficiency, and as a geometric primitive in *Blender* [2023], to ensure that Harnack tracing interoperates as expected with richer rendering features (shadows, reflections, other geometry, *etc.*). In both environments, our algorithm runs fast enough for real-time interaction, with frame rates depending on geometric complexity—but often in excess of 60 fps (see accompanying video). Our GLSL implementations are available in the supplemental material, and our Blender implementation can be found at https://github.com/MarkGillespie/harnack-blender.

Figures 9, 15, 19, 20, 25, and 26 were rendered via GPU shaders. All shaders ran at real time rates on an older GPU (GeForce RTX 3090) at a resolution of 1890 × 1062. Point cloud reconstruction (Figure 15) ran slowest, around 20–30 fps, primarily because we used naïve $O(n)$ evaluation of all dipoles, rather than the $O(\log(n))$ hierarchical evaluation advocated by Barill et al. [2018]. Figures 1, 7, 8, 10, 11, 12, 13, 14, 16, 17, and 27 were rendered in Blender's CPU path tracer, on an Apple M2 Max (8 performance cores, 32GB RAM).

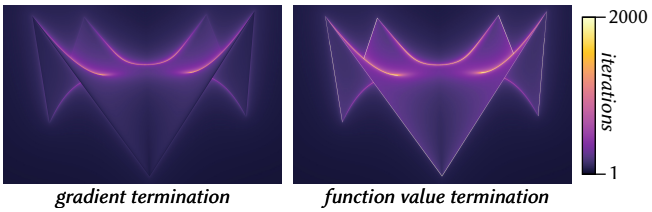


Fig. 22. Here we visualize the number of iterations required to intersect each camera ray with the surface shown in Figure 10. Just as with sphere tracing, Harnack tracing requires the greatest number of steps to find intersections near the shape silhouette, where the camera ray approaches at a glancing angle. We find that the gradient-based termination condition discussed in Section 3.1.2 cuts down on the number of iterations required compared to a naïve termination condition based solely on the function value.

We achieved real-time feedback in Blender's interactive preview (see supplemental video). Final renders took longer: the nonplanar polygons in Figure 8 took around a minute each to render at 1400×1000 resolution, and Figure 1 took five minutes at the same resolution. Architectural surfaces in Figure 17 took 4.5 hours to render, due to the cost of evaluating several elliptic integrals per step. In general, the longest execution times in both CPU and GPU implementations were dominated by evaluating expensive harmonic functions—rather than the logic of Harnack tracing itself. Section 6 discusses acceleration strategies which could cut costs dramatically.

4.9 Tracing Iterations

Figure 22 shows the number of iterations required to intersect each camera ray with a nonplanar polygon via Harnack tracing. Intersections near the silhouette require the most iterations to compute, as the ray approaches the surface at a glancing angle. If one terminates rays based on function value alone (Figure 22, *right*), then points near the polygon boundary require a huge number of iterations since the function's value changes so rapidly in those regions. Using the gradient-based termination condition (Section 3.1.2) dramatically reduces the number of iterations required in those regions, although it may or may not be faster in execution time due to the added cost of evaluating the gradient at each step. In our experiments, we found that the gradient termination condition generally helped in the GLSL implementation, often providing a 10–20% speedup by decreasing the maximum number of iterations required, but often made overall execution times slightly longer in Blender.

4.10 Convergence

We compared the convergence rate of our Harnack scheme with the widely-used sphere tracing algorithm (Section 5.2.2), measuring the number of iterations required to converge near the level set. Since the solid angle function $\Omega_P(\mathbf{x})$ is not Lipschitz, we could not run sphere tracing directly on $\Omega_P(\mathbf{x})$. Instead, we meshed the target level set, and used closest point queries (via *FCPW* [Sawhney 2021]) to run sphere tracing on the resulting mesh. Figure 23 shows convergence plots, both in terms of the error in function value (*i.e.* $|f(\mathbf{r}(t)) - f^*|$), and in terms of absolute distance to the intersection (*i.e.* $|t - t^*|$) at each iteration. Note that this test represents the ideal case for sphere tracing, where the Lipschitz constant is $C = 1$ (*cf.*

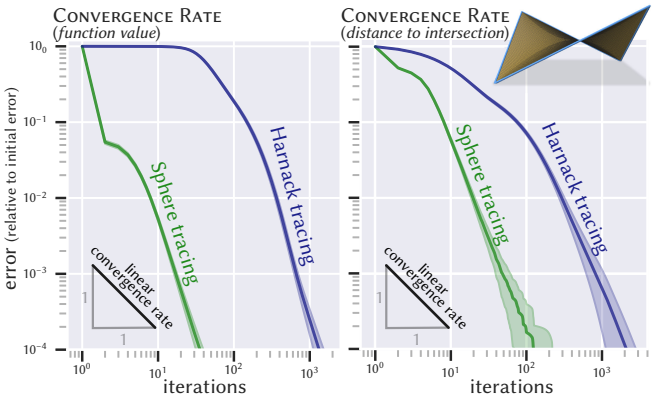


Fig. 23. To compare the convergence rates of Harnack tracing and sphere tracing, we measured the cost of sphere tracing a mesh of the 2π level set of solid angle for a sample curve *(top right)*, as well as the cost of Harnack tracing the level set. The plots show the rate at which the function value $f(\mathbf{r}(t))$ approaches the target value f^* *(left)*, as well as the rate at which the time t approaches the optimal time t^* *(right)*. In practice, both methods converge faster than the linear rate guaranteed in Appendix A, although sphere tracing requires many fewer iterations overall. Center lines show mean error per iteration; shaded regions show 99% confidence intervals.

Figure 26). Empirically, both methods appear to converge faster than the linear rate guaranteed in Appendix A, though sphere tracing still requires significantly fewer iterations than Harnack tracing.

5 RELATED WORK AND COMPARISONS

To our knowledge, there is no prior work that specifically considers ray tracing algorithms for harmonic functions (apart from general-purpose techniques like ray marching or root finding). In general, implicit surface visualization has been studied for centuries in mathematics [Wallis 1659]; and used for decades in computer graphics [Goldstein and Nagel 1971] (see [Knoll 2007] for a survey). Here there are several basic classes of techniques:

- **Explicit Conversion.** The implicit surface is converted to an explicit mesh via an isosurface extraction method such as *marching cubes* [Lorensen and Cline 1987]; this mesh is then rasterized. While rasterization is fast, isosurface extraction is compute- and memory-intensive, making it ill-suited to dynamic geometry or per-pixel accurate rendering. Standard isosurfacing algorithms are also not well-adapted to the angle-valued, singular nature of our harmonic functions, as examined in Section 5.1.
- **Intermediate Representation.** The implicit surface is converted to an intermediate representation such as a volume density, which is still visualized via ray tracing [Hadwiger et al. 2005]. As noted by Knoll [2007], this approach requires a very sharp transfer function to render hard surfaces, which can incur aliasing. Moreover, though it may reduce bandwidth in certain settings (*e.g.*, visualizing dense point clouds), Harnack tracing will be far more cache friendly for the kinds of compact surface descriptions considered in Section 4 (which in many cases can fit entirely into registers).
- **Ray Tracing.** By far the most common technique for rendering smooth implicit surfaces—which we also adopt—is to compute

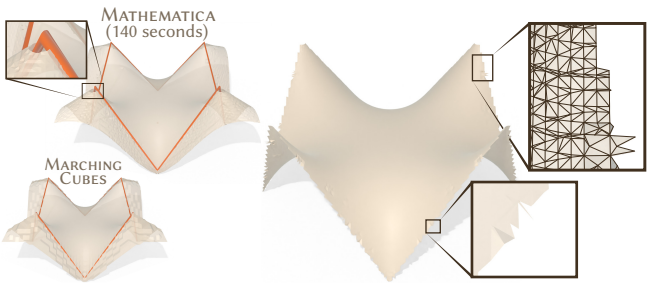


Fig. 24. Simply running marching cubes "out of the box" on many of our example problems yields surfaces with unsightly artifacts, especially around the jump discontinuities present when f is an angle-valued function *(bottom left)*. More sophisticated adaptive methods, like Mathematica's `ContourPlot3D`, suffer from similar artifacts and also struggle near the singularities at the vertices of the polygon *(top left)*. One can attempt to filter out the extraneous faces by, *e.g.* evaluating f at the barycenter of each face, but doing so still leaves behind a noisy surface near singularities *(right)*.

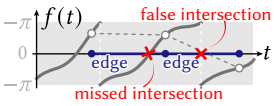
ray intersections (Section 5.2). A basic strategy is uniform *ray marching*, which is asymptotically slower than Harnack tracing, and yields significant visual artifacts (Section 5.2.1). A more sophisticated class of strategies are *sphere tracing algorithms*, which use bounds on the gradient, or more general *Lipschitz bounds*, to take large but conservative steps. The fundamental problem here is that, in general, harmonic functions are simply not Lipschitz (Section 5.2.2). Since ray tracing restricts the implicit function to an ordinary 1D function, one can also identify intersections via generic *root finding* algorithms such as Newton's method or bisection search. For harmonic functions, however, these methods are not guaranteed to find the closest root (Section 5.3).

- **Interval Analysis.** To obtain stronger guarantees, one can apply *interval analysis* techniques—either along a ray [Knoll et al. 2009; Sharp and Jacobson 2022], or over a voxelization of space [Keeter 2020]. These methods provide conservative bounds on implicit function values, but can sometimes be *too* conservative (and hence costly). Existing interval analysis techniques also do not apply to angle-valued functions, producing incorrect results for many of the surfaces we consider (Figure 27, *bottom right*).

Overall, despite the vast literature on implicit surface visualization, the types of functions we consider (namely, singular angle-valued harmonic functions) pose challenges of efficiency and/or robustness for essentially all prior classes of algorithms, which treat them as literal discontinuous functions. Harnack tracing instead views these functions correctly as a continuation of a global harmonic function (Section 3.2), yielding better-behaved results: *e.g.* higher-quality results for equal compute time. Of course, one could try to extend *any* of the strategies above to the general harmonic case—an interesting question we leave to future work.

5.1 Explicit Conversion

Explicit meshing approximates a level set of a function f by evaluating f at the nodes of a fixed [Lorensen and Cline 1987; Ju et al. 2002] or dynamic



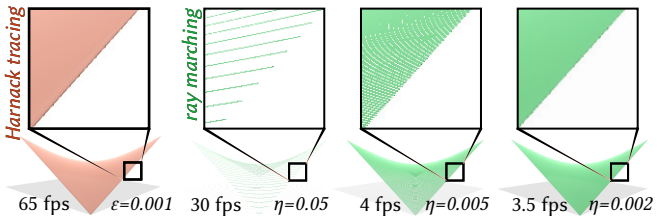
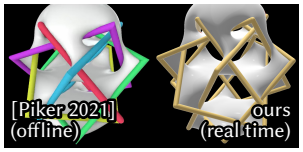


Fig. 25. Ray marching with a fixed step size η is a simple way of ray tracing implicit surfaces. However, using large step sizes leads to severe artifacts, as the algorithm can easily "tunnel" through the surface, while small values of η quickly become prohibitively expensive. (Images rendered at 1423×800.)

[Shen et al. 2023] background grid; the basic idea is to place mesh vertices where edges cross the level set of f. However, if f is angle-valued this strategy can fail—in the inset, *e.g.*, simply comparing function values at edge endpoints yields both false negatives and false positives. Moreover, such methods struggle to resolve geometry near singularities—even with adaptive refinement (Figure 24).



To date, there appears to be no established isosurfacing method for angle-valued functions—Piker [2021] demonstrates an unpublished, undocumented method which works well for our problem, but does not achieve real time frame rates as we do with Harnack tracing. Such methods are especially impractical for achieving per-pixel accuracy. In general, our experience has been that ray tracing is necessary for fast, high-quality visualization of harmonic functions.

5.2 Ray Tracing

For an implicit surface $\mathcal{S}$ defined by a function $f(\mathbf{x})$ (*à la* Equation 2), and a ray $\mathbf{r}(t)$ (Equation 3), ray tracing algorithms seek times $t > 0$ at which the composite function $f(\mathbf{r}(t))$ equals a target value f^*, or equivalently, roots (*i.e.*, zeros) of the function

$$\phi(t) = f(\mathbf{r}(t)) - f^*. \tag{20}$$

In general, it is hard to guarantee that ray tracing finds the smallest such t value, without making special assumptions about f. In fact, there are only a few classes of surfaces that provide a first-hit guarantee. A reasonably comprehensive list includes planar polygons [Appel 1968], quadrics [Goldstein and Nagel 1971], (piecewise) algebraic surfaces [Hanrahan 1983] such as tori [Roth 1982], metaballs [Tatsumi et al. 1990], algebraic swept surfaces [van Wijk 1985], superquadrics [Barr 1981; Edwards 1982], Lipschitz functions [Kalra and Barr 1989; Hart 1996] (including signed distance functions [Hart et al. 1989]), some subdivision surfaces [Kobbelt et al. 1998], and CSG hierarchies built from any of these primitives [Goldstein and Nagel 1971]. Other implicit functions, defined *e.g.* by point sets [Adamson and Alexa 2003] or neural fields [Takikawa et al. 2023], may need to be ray traced via general-purpose root finding techniques—which may sometimes fail to find any intersection, much less the first hit. For further discussion, see surveys by Hanrahan [1989] and Hart [1993a], as well as more recent overviews in Galin et al. [2020, §2] and Aydinlilar and Zanni [2021, §2].

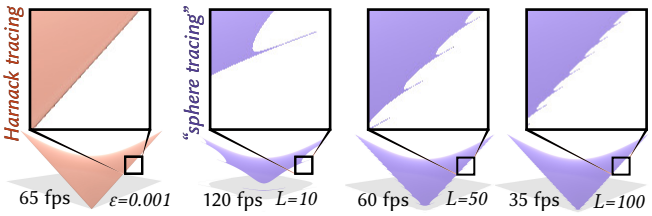


Fig. 26. Though many of our harmonic functions are not Lipschitz, we can run sphere tracing using any purported Lipschitz bound L. However, when working with functions like the solid angle where no valid Lipschitz bound exists, this technique yields artifacts near singularities, even for high values of L where the performance also degrades. (Images rendered at 1423×800.)

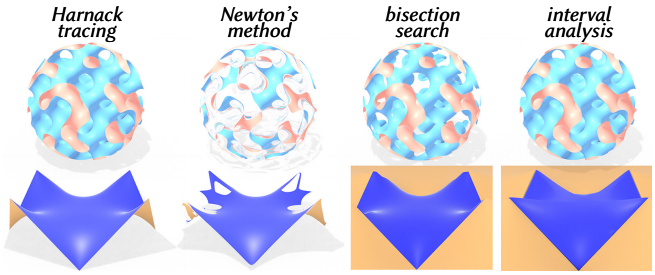


Fig. 27. General-purpose root finding techniques may fail to identify the *closest* point of intersection, yielding incorrect occlusions or even gaps in the surface if they converge to intersections outside of the visible area. Such problems are especially severe on the angle-valued functions used in many of our examples, where bisection search and interval analysis erroneously interpret jump discontinuities as roots. These spurious intersections form opaque backgrounds in the renders, occluding parts of the polygon which should be visible behind the jump discontinuity *(bottom, right)*.

5.2.1 Ray Marching. Ray marching with a fixed step size is perhaps the simplest way to ray trace an implicit surface, but suffers from fundamental flaws. First, as the stopping tolerance $\varepsilon \to 0$ ray marching is asymptotically slower than methods based on conservative bounds: ray marching takes $O(1/\varepsilon)$ steps, whereas Harnack tracing takes only $O(\log(1/\varepsilon))$ steps (Appendix A). Second, for large step sizes ray marching can easily "tunnel" through the level set, leaving gaps in the surface. And in practice, we find no satisfactory trade off: ray marching either produces unacceptable artifacts, or runs orders of magnitude slower than Harnack tracing (Figure 25).

5.2.2 Sphere Tracing.

"Sphere tracing likes distance, and avoids nothing."
—John C. Hart [1993]

As described in Section 1, *sphere tracing* is a strategy for efficiently ray tracing functions with a known Lipschitz bound. It was first developed by Hart *et al.*, who describe how to visualize quaternionic and linear fractals, algebraic surfaces, CSG composites with hard or soft blends, and spatial deformations [Hart et al. 1989; Hart and DeFanti 1991; Hart 1993b, 1996]. Quilez [2008] significantly expanded this list, and popularized SDF-based sphere tracing by demonstrating its ability to render complex scenes. Crane [2005] describes the first GPU implementation, and recent work explores differentiable versions suitable for machine learning [Liu et al. 2020; Jiang et al. 2020; Bangaru et al. 2022; Vicini et al. 2022]. Seyb et al. [2019] develop an efficient strategy for sphere tracing large deformations, and Galin et al. [2020] describe a strategy for computing local Lipschitz bounds along a ray, for certain classes of composite shapes—unfortunately, this class does not include harmonic level sets. There has also been recent work on accelerating sphere tracing [Keinert et al. 2014; Bálint and Valasek 2018]. We use one of these strategies (over-stepping) to accelerate Harnack tracing in Section 3.1.4; others (such as multiresolution rendering) could likely yield further accelerations. Likewise, one could easily adopt the *cone tracing* strategy of Hart [1993b, §4.3] for anti-aliasing.

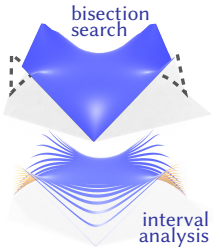
Unfortunately, although harmonic functions are extremely regular, they are not globally Lipschitz—which is why the Harnack inequality takes the form that it does, *i.e.*, going to infinity at the boundary of a finite ball. Moreover, many of the particular harmonic functions that we are interested in, such as the solid angle associated with a polygon, may have no local Lipschitz bound even within small neighborhoods—*e.g.*, the solid angle can change arbitrarily fast near a curve (Section 4.2). Hence, existing sphere tracing/Lipschitz methods cannot directly provide intersection guarantees for harmonic functions. Nonetheless, we can still attempt to apply traditional sphere tracing by taking a large value L as a purported Lipschitz bound. As shown in Figure 26, however, we end up with either severe artifacts when L is small, or pay a large cost when L is large—and still do not completely eliminate artifacts around the curve.

Figure 26 also shows that in this test case, Harnack tracing has a similar computational cost to sphere tracing with a Lipschitz constant of $L = 50$, which is significantly more expensive than running sphere tracing for lower values of L. So if one has a function with a small known Lipschitz bound, then sphere tracing will probably be more efficient; the real strength of Harnack tracing is that it can be applied to functions for which Lipschitz bounds do not exist, like solid angle. But of course, the dominant computational cost in both Harnack tracing and sphere tracing is generally in evaluating the function f, rather than evaluating the step size, so one could always compute step sizes using both sphere tracing and Harnack tracing and take the larger of the two, since both are guaranteed to be safe.

5.3 Root Finding

Ray tracing can also be viewed as a root finding problem for the function $\phi(t)$ (Equation 20). In special situations like Section 4.1, where $f(\mathbf{x})$ is a polynomial, one can apply polynomial root finding methods (like *Sturm sequences*) to compute *all* roots of $\phi(t)$, and take the smallest one. More commonly, however, one must turn to general-purpose root-finding methods. For instance, *Newton's method* iteratively finds the root of a local linear approximation; *bisection search* iteratively splits an interval that straddles a root, and similarly, *interval analysis* tracks the upper and lower bounds of an interval, which is recursively split until it has width no greater than a tolerance ε around a root, or is guaranteed not to contain one. Even if these methods guarantee *some* root is found, they can

in general fail to find the *first* hit (*i.e.*, the smallest positive root t), leading to occlusion artifacts or gaps in the surface.



Moreover, for angle-valued functions, one may detect discontinuities in $f(\mathbf{x})$, rather than true geometric intersections. For interval-based methods, one can try to "patch" this issue by, *e.g.*, checking whether the value of $\phi(t)$ at the center of the interval is within ε of zero, but this sort of modification voids any guarantees—causing new artifacts (see inset). For level sets of simple functions, like the globally continuous gyroid in Figure 27 *(top right)*, interval analysis can reliably compute the first intersection. But for the broader class of surfaces handled by Harnack tracing, significant artifacts were visible in all root finding methods we tried (Figure 27).

5.4 Walk on Spheres

Finally, our method has some potentially fruitful connections to the *walk on spheres (WoS)* method [Muller 1956], recently used in graphics to solve partial differential equations (PDEs) in a variety of settings [Sawhney and Crane 2020; Sawhney et al. 2022, 2023; Miller et al. 2023; Bakbouk and Peers 2023; Yılmazer et al. 2022; Rioux-Lavoie et al. 2022; Sugimoto et al. 2023; Miller et al. 2024]. First, WoS can solve PDEs on any domain where one can determine conservative empty spheres—hence, the strategies we develop to evaluate Harnack bounds in Section 4 extend WoS to a richer set of geometries. Conversely, WoS can be used to obtain pointwise evaluations of harmonic functions on arbitrary geometric domains, providing an even larger set of functions that can be visualized via Harnack tracing. However, despite the close connection to spheres and harmonic functions, WoS is not an *alternative* to Harnack tracing—they are different algorithms, with fundamentally different purposes (evaluating a harmonic function, versus visualizing its level sets).

6 LIMITATIONS AND FUTURE WORK

As noted in Section 4.10, Harnack tracing is less efficient than sphere tracing—though it at least exhibits the same *asymptotic* rate of convergence. As hinted at in Section 5, there are many opportunities to close the gap. For instance, a common acceleration for sphere tracing, which we did not implement, is to set the stopping tolerance adaptively according to the pixel size and distance from the camera. Another useful trick is to use low-resolution cone tracing to provide safe initialization to ray trace a higher-resolution image [Bálint and Valasek 2018]. More fundamentally, one could try to derive tighter bounds by additionally incorporating, *e.g.*, the ray direction or gradient information. Specific applications can also be significantly accelerated: *e.g.*, one could visualize Poisson reconstructed surfaces dramatically faster by applying Barnes Hut/fast multipole methods, *à la* Barill et al. [2018]—exponentially reducing the cost of evaluating the harmonic function at each step.

The present work considers the Harnack inequality for harmonic functions, but Harnack inequalities are a much more general object studied in elliptic PDE theory—hinting that there may also be efficient ray tracing algorithms for a much broader class of surfaces [Kassmann 2007]. Similarly, just as the SDF condition $\|\nabla\phi\| = 1$ can be relaxed to the Lipschitz condition $\|\nabla\phi\| \leq 1$, enabling a broader class of objects to be ray traced via classic sphere tracing, it may be possible to transition from strictly harmonic functions $\Delta u = 0$ to *subharmonic* functions $\Delta u \leq 0$ to expand the class of surfaces where Harnack tracing can be applied. In general, our feeling is that the Harnack approach holds significant potential for expanding the class of surfaces that can be efficiently and reliably ray traced.

Finally, our exploration of this topic was motivated in part by recent interest in SDFs as a neural surface representation [Xie et al. 2022]. Sphere tracing has experienced a renaissance as one of the basic strategies for visualizing neural SDFs [Takikawa et al. 2021, §2]. However, vision and learning continue to seek alternative surface representations, in part due to challenges with enforcing the signed distance property. To ensure that a function f is an SDF, one must apply nonconvex, nondifferentiable loss functions to enforce the *eikonal condition* $\|\nabla f\| = 1$ [Gropp et al. 2020]. Moreover, it is well-known that this condition may still fail to provide the signed distance property [Xie et al. 2022], which is more difficult to enforce directly [Marschner et al. 2023]. In contrast, ensuring that f is harmonic amounts to a simple linear condition $\Delta f = 0$; it also corresponds to a convex *Dirichlet energy* $\int \|\nabla f\|^2$, suggesting it may be easier to optimize. However, optimizing "neural harmonic surfaces" would require a differentiable version of Harnack tracing—which we leave to future work (and may play well with recent angle-valued neural representations [Palmer et al. 2022]).

ACKNOWLEDGMENTS

Thanks to Nicole Feng for help with Biot-Savart-based expressions for our surface normals, to Albert Chern for helpful discussion of the Biot-Savart law and random projections of curves, and to Andrea Tagliasacchi and Google Brain for supporting this work. Figure 2 is based on an illustration by the last author [Crane 2012]. This work was funded by an NSF CAREER Award (IIS 1943123), NSF Award IIS 2212290, a Packard Fellowship and a gift from Google Brain.

REFERENCES

Anders Adamson and Marc Alexa. 2003. Approximating and intersecting surfaces from points. In *Symposium on Geometry Processing*. 230–239. https://doi.org/10.5555/882370.882401

Emil Adiels, Mats Ander, and Chris JK Williams. 2022. The architectural application of shells whose boundaries subtend a constant solid angle. *arXiv preprint* (2022), 21. https://arxiv.org/pdf/2212.05913.pdf

Arthur Appel. 1968. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*. ACM, 37–45. https://doi.org/10.1145/1468075.1468082

Sheldon Axler, Paul Bourdon, and Ramey Wade. 2013. *Harmonic Function Theory* (2nd ed.). Graduate Texts in Mathematics, Vol. 137. Springer. https://doi.org/10.1007/978-1-4757-8137-3

Melike Aydinlilar and Cedric Zanni. 2021. Fast Ray Tracing of Scale-Invariant Integral Surfaces. *Computer Graphics Forum* 40, 6, 117–134. https://doi.org/10.1111/cgf.14208

Ghada Bakbouk and Pieter Peers. 2023. Mean Value Caching for Walk on Spheres. In *Eurographics Symposium on Rendering*. The Eurographics Association, 10. https://doi.org/10.2312/sr.20231120

Csaba Bálint and Gábor Valasek. 2018. Accelerating Sphere Tracing. In *Eurographics (Short Papers)*. 29–32. https://doi.org/10.2312/egs.20181037

Sai Praveen Bangaru, Michaël Gharbi, Fujun Luan, Tzu-Mao Li, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. 2022. Differentiable Rendering of Neural SDFs through Reparameterization. In *SIGGRAPH Asia 2022 Conference Papers*. 1–9. https://doi.org/10.1145/3550469.3555397

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. 2018. Fast Winding Numbers for Soups and Clouds. *ACM Transactions on Graphics (TOG)* 37, 4, Article 43 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201337

Alan H. Barr. 1981. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications* 1, 1 (1981), 11–23. https://doi.org/10.1109/MCG.1981.1673799

Jack Binysh and Gareth P Alexander. 2018. Maxwell's theory of solid angle and the construction of knotted fields. *Journal of Physics A: Mathematical and Theoretical* 51, 38 (2018), 21. https://doi.org/10.1088/1751-8121/aad8c6

Astrid Bunge, Marc Alexa, and Mario Botsch. 2023. Discrete Laplacians for General Polygonal and Polyhedral Meshes. In *SIGGRAPH Asia 2023 Courses*. 1–49. https://doi.org/10.1145/3610538.3614620

Astrid Bunge, Philipp Herholz, Misha Kazhdan, and Mario Botsch. 2020. Polygon Laplacian made simple. *Computer Graphics Forum* 39, 2 (2020), 303–313. https://doi.org/10.1111/cgf.13931

Edwin E. Catmull and James H. Clark. 1978. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design* 10, 6 (1978), 350–355. https://doi.org/10.1016/0010-4485(78)90110-0

Albert Chern and Sadashige Ishida. 2023. Area formula for spherical polygons via prequantization. *arXiv preprint* (2023), 12. https://arxiv.org/pdf/2303.14555.pdf

Keenan Crane. 2005. Ray Tracing Quaternion Julia Sets on the GPU. https://www.cs.cmu.edu/~kmcrane/Projects/QuaternionJulia/.

Keenan Crane. 2012. Graph of Harnack's inequality. https://commons.wikimedia.org/w/index.php?title=File:Graph_of_Harnack%27s_inequality.png&oldid=600951383 [Online; accessed 27-March-2024].

Olaf Diegel. 2021. Design for Additive Manufacturing: A workflow for a metal AM heat exchanger using nTopology. *Metal AM* 7, 2 (2021), 185–189.

Bruce E Edwards. 1982. *Implementation of a ray-tracing algorithm for rendering superquadric solids*. Rensselaer Polytechnic Institute, Troy, NY.

Michael S. Floater. 2003. Mean Value Cordinates. *Computer Aided Geometric Design* 20, 1 (2003), 19–27. https://doi.org/10.1016/S0167-8396(03)00002-5

Blender Foundation and Community. 2023. *Blender 4.0*. http://www.blender.org

Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Patrick Alken, Michael Booth, and Fabrice Rossi. 2009. *GNU Scientific Library Reference Manual* (3rd ed.). Network Theory Ltd.

Eric Galin, Eric Guérin, Axel Paris, and Adrien Peytavie. 2020. Segment tracing using local Lipschitz bounds. *Computer Graphics Forum* 39, 2 (2020), 545–554. https://doi.org/10.1111/cgf.13951

Fernando de Goes, Siome Goldenstein, Mathieu Desbrun, and Luiz Velho. 2011. Exoskeleton: Curve Network Abstraction for 3D Shapes. *Computers & Graphics* 35, 1 (Feb. 2011), 112–121. https://doi.org/10.1016/j.cag.2010.11.012

Robert A. Goldstein and Roger Nagel. 1971. 3-D Visual simulation. *Simulation* 16, 1 (1971), 25–31. https://doi.org/10.1177/003754977101600104

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Vol. 119. PMLR, 3789–3799. https://proceedings.mlr.press/v119/gropp20a.html

Markus Hadwiger, Christian Sigg, Henning Scharsach, Khatja Bühler, and Markus Gross. 2005. Real-time ray-casting and advanced shading of discrete isosurfaces. *Computer Graphics Forum* 24, 3 (2005), 303–312. https://doi.org/10.1111/j.1467-8659.2005.00855.x

Pat Hanrahan. 1983. Ray Tracing Algebraic Surfaces. *SIGGRAPH Computer Graphics* 17, 3 (July 1983), 83–90. https://doi.org/10.1145/964967.801136

Pat Hanrahan. 1989. A Survey of Ray-Surface Intersection Algorithms. In *An Introduction to Ray Tracing*, Andrew S. Glassner (Ed.). Academic Press, 79–119.

Axel Harnack. 1887. *Die Grundlagen der Theorie des logarithmischen Potentiales und der eindeutigen Potentialfunktion in der Ebene*. VG Teubner.

John C. Hart. 1993a. Ray Tracing Implicit Surfaces. In *Modeling, Visualizing, and Animating Implicit Surfaces (Siggraph 1993 Courses)*.

John C Hart. 1993b. Sphere tracing: Simple robust antialiased rendering of distance-based implicit surfaces. In *SIGGRAPH*, Vol. 93. 1–11.

John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545. https://doi.org/10.1007/s003710050084

John C Hart and Thomas A DeFanti. 1991. Efficient antialiased rendering of 3-D linear fractals. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. 91–100. https://doi.org/10.1145/122718.122728

John C. Hart, Daniel J. Sandin, and Louis H. Kauffman. 1989. Ray tracing deterministic 3-D fractals. *SIGGRAPH Computer Graphics* 23, 3 (July 1989), 289–296. https://doi.org/10.1145/74334.74363

Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics (TOG)* 32, 4 (July 2013), 1–12. https://doi.org/10.1145/2461912.2461916

Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. 2020. SDFDiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1251–1261.

Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. 2007. Harmonic Coordinates for Character Articulation. *ACM Transactions on Graphics (TOG)* 26, 3 (July 2007). https://doi.org/10.1145/1276377.1276466

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. 2002. Dual contouring of Hermite data. *ACM Transactions on Graphics (TOG)* 21, 3 (July 2002), 339–346. https://doi.org/10.1145/566654.566586

Devendra Kalra and Alan H. Barr. 1989. Guaranteed Ray Intersections with Implicit Surfaces. *SIGGRAPH Computer Graphics* 23, 3 (July 1989), 297–306. https://doi.org/10.1145/74334.74364

Moritz Kassmann. 2007. Harnack Inequalities: an Introduction. *Boundary Value Problems* 2007 (2007), 1–21. https://doi.org/10.1155/2007/81415

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson Surface Reconstruction. In *Symposium on Geometry Processing*. Eurographics Association, 61–70. https://doi.org/10.2312/SGP/SGP06/061-070

Matthew J. Keeter. 2020. Massively parallel rendering of complex closed-form implicit surfaces. *ACM Transactions on Graphics (TOG)* 39, 4, Article 141 (Aug. 2020), 10 pages. https://doi.org/10.1145/3386569.3392429

Benjamin Keinert, Henry Schäfer, Johann Korndörfer, Urs Ganse, and Marc Stamminger. 2014. Enhanced Sphere Tracing. In *Smart Tools and Apps for Graphics*. The Eurographics Association. https://doi.org/10.2312/stag.20141233

Alois Knoll. 2007. A survey of implicit surface rendering methods, and a proposal for a common sampling framework. In *Proceedings of the 2nd IRTG Workshop (GI Lecture Notes in Informatics)*.

Aaron Knoll, Younis Hijazi, Andrew Kensler, Mathias Schott, Charles Hansen, and Hans Hagen. 2009. Fast ray tracing of arbitrary implicit surfaces with interval and affine arithmetic. *Computer Graphics Forum* 28, 1 (2009), 26–40. https://doi.org/10.1111/j.1467-8659.2008.01189.x

Leif P Kobbelt, Katja Daubert, and Hans-Peter Seidel. 1998. Ray tracing of subdivision surfaces. In *Eurographics Workshop on Rendering Techniques*. Springer, 69–80. https://doi.org/10.1007/978-3-7091-6453-2_7

John M. Lee. 2018. *Introduction to Riemannian Manifolds* (2nd ed.). Graduate Texts in Mathematics, Vol. 176. Springer. https://doi.org/10.1007/978-3-319-91755-9

Adrien Marie Legendre. 1817. *Éléments de géometrie* (11th ed.). translated as: *Elements of Geometry* (1819). Cambridge University Press.

Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. 2020. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019–2028. https://doi.org/10.1109/CVPR42600.2020.00209

William E. Lorensen and Harvey E. Cline. 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *SIGGRAPH Computer Graphics* 21, 4 (Aug. 1987), 163–169. https://doi.org/10.1145/37402.37422

Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. 2023. Constructive Solid Geometry on Neural Signed Distance Fields. In *SIGGRAPH Asia 2023 Conference Papers*. ACM, Article 121, 12 pages. https://doi.org/10.1145/3610548.3618170

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2023. Boundary Value Caching for Walk on Spheres. *ACM Transactions on Graphics (TOG)* 42, 4 (July 2023). https://doi.org/10.1145/3592400

Bailey Miller, Rohan Sawhney, Keenan Crane, and Ioannis Gkioulekas. 2024. Walkin' Robin: Walk on Stars with Robin Boundary Conditions. *ACM Transactions on Graphics (TOG)* 43, 41 (2024).

Mervin E. Muller. 1956. Some Continuous Monte Carlo Methods for the Dirichlet Problem. *Annals of Mathematical Statistics* 27, 3 (1956), 569–589. https://doi.org/10.1214/aoms/1177728169

Adriaan van Oosterom and Jan Strackee. 1983. The solid angle of a plane triangle. *IEEE Transactions on Biomedical Engineering* BME-30, 2 (1983), 125–126. https://doi.org/10.1109/TBME.1983.325207

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics (TOG)* 27, 3 (Aug. 2008), 1–8. https://doi.org/10.1145/1360612.1360691

David Palmer, Dmitriy Smirnov, Stephanie Wang, Albert Chern, and Justin Solomon. 2022. DeepCurrents: Learning Implicit Representations of Shapes with Boundaries. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18644–18654. https://doi.org/10.1109/CVPR52688.2022.01811

Frank Paxton. 1959. Solid angle calculation for a circular disk. *Review of Scientific Instruments* 30, 4 (1959), 254–258. https://doi.org/10.1063/1.1716590

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2023. *Physically based rendering: From theory to implementation* (4th ed.). Morgan Kaufmann. https://www.pbrt.org/

Daniel Piker. 2021. Some examples of periodic isosurfacing. https://twitter.com/KangarooPhysics/status/1457802855994191877.

Ulrich Pinkall and Konrad Polthier. 1993. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics* 2, 1 (1993), 15–36. https://doi.org/10.1080/10586458.1993.10504266

Inigo Quilez. 2008. Raymarching Signed Distance Fields. https://iquilezles.org/articles/raymarchingdf/. Accessed: 2023-06-20.

Inigo Quilez. 2015. Normals for an SDF. https://iquilezles.org/articles/normalsSDF/. Accessed: 2023-06-20.

Inigo Quilez and Pol Jeremias. 2013. ShaderToy. https://www.shadertoy.com/. Accessed: 2024-01-21.
Damien Rioux-Lavoie, Ryusuke Sugimoto, Tümay Özdemir, Naoharu H. Shimada, Christopher Batty, Derek Nowrouzezahrai, and Toshiya Hachisuka. 2022. A Monte Carlo Method for Fluid Simulation. *ACM Transactions on Graphics (TOG)* 41, 6, Article 240 (Nov. 2022), 16 pages. https://doi.org/10.1145/3550454.3555450
Scott D. Roth. 1982. Ray casting for modeling solids. *Computer Graphics and Image Processing* 18, 2 (1982), 109–144. https://doi.org/10.1016/0146-664X(82)90169-1
Robert E. Rothe. 1969. The solid angle at a point subtended by a circle. *Journal of the Franklin Inst.* 287, 6 (1969), 515–521. https://doi.org/10.1016/0016-0032(69)90061-1
Rohan Sawhney. 2021. *FCPW: Fastest Closest Points in the West.* https://github.com/rohan-sawhney/fcpw
Rohan Sawhney and Keenan Crane. 2020. Monte Carlo Geometry Processing: A Grid-Free Approach to PDE-Based Methods on Volumetric Domains. *ACM Transactions on Graphics (TOG)* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392374
Rohan Sawhney, Bailey Miller, Ioannis Gkioulekas, and Keenan Crane. 2023. Walk on Stars: A Grid-Free Monte Carlo Method for PDEs with Neumann Boundary Conditions. *ACM Transactions on Graphics (TOG)* 42, 4 (2023). https://doi.org/10.1145/3592398
Rohan Sawhney, Dario Seyb, Wojciech Jarosz, and Keenan Crane. 2022. Grid-Free Monte Carlo for PDEs with Spatially Varying Coefficients. *ACM Transactions on Graphics (TOG)* 41, 4 (July 2022). https://doi.org/10.1145/3528223.3530134
Dario Seyb, Alec Jacobson, Derek Nowrouzezahrai, and Wojciech Jarosz. 2019. Non-linear sphere tracing for rendering deformed signed distance fields. *ACM Transactions on Graphics (TOG)* 38, 6, Article 229 (Nov. 2019), 12 pages. https://doi.org/10.1145/3355089.3356502
Nicholas Sharp and Alec Jacobson. 2022. Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis. *ACM Transactions on Graphics (TOG)* 41, 4, Article 107 (July 2022), 16 pages. https://doi.org/10.1145/3528223.3530155
Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. 2023. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Transactions on Graphics (TOG)* 42, 4 (July 2023), 1–16. https://doi.org/10.1145/3592430
William R. Smythe. 1989. *Static and Dynamic Electricity* (3rd ed.). Taylor & Francis. revised printing.
Ryusuke Sugimoto, Terry Chen, Yiti Jiang, Christopher Batty, and Toshiya Hachisuka. 2023. A Practical Walk-on-Boundary Method for Boundary Value Problems. *ACM Transactions on Graphics (TOG)* 42, 4, Article 81 (July 2023), 16 pages. https://doi.org/10.1145/3592109
Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 11358–11367. https://doi.org/10.1109/CVPR46437.2021.01120
Towaki Takikawa, Shunsuke Saito, James Tompkin, Vincent Sitzmann, Srinath Sridhar, Or Litany, and Alex Yu. 2023. Neural Fields for Visual Computing. In *ACM SIGGRAPH 2023 Courses.*
Haruyuki Tatsumi, Eiji Takaoki, Koichi Omura, and Hisao Fujita. 1990. A new method for three-dimensional reconstruction from serial sections by computer graphics using "meta-balls": Reconstruction of "hepatoskeletal system" formed by Ito cells in the cod liver. *Computers and Biomedical Research* 23, 1 (1990), 37–45. https://doi.org/10.1016/0010-4809(90)90005-W
Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *ACM Transactions on Graphics (TOG)* 41, 4 (July 2022), 1–18. https://doi.org/10.1145/3528223.3530139
John Wallis. 1659. *Tractatus de Sectionibus Conicis.*
Stephanie Wang and Albert Chern. 2021. Computing minimal surfaces with differential forms. *ACM Transactions on Graphics (TOG)* 40, 4 (July 2021), 1–14. https://doi.org/10.1145/3450626.3459781
Jarke J. van Wijk. 1985. Ray tracing objects defined by sweeping a sphere. *Computers & Graphics* 9, 3 (1985), 283–290. https://doi.org/10.1016/0097-8493(85)90055-X
Jarke J. van Wijk and Arjeh M. Cohen. 2006. Visualization of Seifert Surfaces. *IEEE IEEE Transactions on Visualization and Computer Graphics* 12, 4 (2006), 485–496. https://doi.org/10.1109/TVCG.2006.83
Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural fields in visual computing and beyond. *Computer Graphics Forum* 41, 2 (2022), 641–676. https://doi.org/10.1111/cgf.14505
Ekrem Fatih Yılmazer, Delio Vicini, and Wenzel Jakob. 2022. Solving Inverse PDE Problems using Grid-Free Monte Carlo Estimators. *arXiv preprint* (2022). https://arxiv.org/pdf/2208.02114.pdf

A CONVERGENCE ANALYSIS

In Theorem A.1, we show that Harnack tracing converges linearly—the same convergence rate offered by sphere tracing. Explicitly, this means that as the stopping tolerance $\varepsilon \to 0$, Harnack tracing will require $O(\log(1/\varepsilon))$ steps to find a time t such that $|f(\mathbf{r}(t)) - f^*| < \varepsilon$.

It is perhaps not obvious, *a priori*, that Harnack tracing must converge to an intersection with the target level set $\mathcal{S} \coloneqq f^{-1}(f^*)$. One might worry about a pathological choice of ball radii R and lower bounds c leading to an ever decreasing sequence of steps and "getting stuck" at some point in space away far from $\mathcal{S}$. Indeed, in Section 4.1, we saw that if the ball radius R becomes zero in some region of the domain, then Harnack tracing will be unable to step past that region. But Theorem A.1 shows that this is the only issue: so long as the radii R are nonzero, then any point at which Harnack tracing "gets stuck" must belong to the level set $\mathcal{S}$.

Theorem A.1. *Suppose the radius $R(\mathbf{x})$ and shift $c(\mathbf{x})$ are compatible, in the sense that $f(\mathbf{x}) - c(\mathbf{x}_0) > 0$ on the ball $B_{R(\mathbf{x}_0)}(\mathbf{x}_0)$ for all $\mathbf{x}_0$, and that $R(\mathbf{x}) > 0$. Then Harnack tracing converges linearly to the first intersection of $\mathbf{r}(t)$ with $\mathcal{S}$, so long as an intersection exists.*

Proof. We divide the proof into two lemmas: Lemma A.2 establishes convergence to the first intersection, while Lemma A.3 establishes the convergence rate. □

Lemma A.2. *Under the assumptions of Theorem A.1, Harnack tracing converges to the first intersection of $\mathbf{r}(t)$ with the surface $\mathcal{S}$.*

Proof. Suppose that the algorithm converges to some time t^*. Since the step size ρ depends continuously on f, R, and c, which in turn depend continuously on position, the step size at position $\mathbf{r}(t^*)$ must be zero. Since R is nonzero, Line 13 of Algorithm 1 implies that we must have $a = 0$, which by Line 12 implies that $f(\mathbf{r}(t^*)) = f^*$. Hence, Harnack tracing converges to an intersection of ray $\mathbf{r}(t)$ with the desired level set (or at least its closure). Harnack tracing can never step past $\mathcal{S}$, since the Harnack inequality guarantees that $f(\mathbf{r}(t))$ stays above/below f^* during each step, so this must be the first intersection between $\mathbf{r}(t)$ and $\mathcal{S}$. □

Lemma A.3. *Under the assumptions of Theorem A.1, Harnack tracing converges linearly.*

Proof. Suppose that Harnack tracing converges to some time t^*. We fix a small neighborhood U of $\mathbf{r}(t^*)$ and define constants $c_U \coloneqq \inf_{\mathbf{x}\in U} c(\mathbf{x})$ and $R_U \coloneqq \inf_{\mathbf{x}\in U} R(\mathbf{x})$. Since our step size increases monotonically with c and R, we can bound the convergence rate by assuming that c and R take on the constant values c_U and R_U within the neighborhood U. Furthermore, by making U sufficiently small we may assume that $f(\mathbf{x})$ is a linear function, and we can absorb the shift c_U into its constant term, leaving us with a positive function $f(\mathbf{x})$ and a positive level set value $f^* > 0$. By shrinking U again if necessary we can also ensure that $|f(\mathbf{x}) - f^*| \leq \frac{1}{2} f^*$ on U.

Hence, without loss of generality we may consider a linear function $f(\mathbf{x}) = \alpha x_2 + \beta$ with constant ball radius R_U. At each iteration, our error is given by $|\alpha x_2 + \beta - f^*|$, and we take a step of size

$$\rho \coloneqq \tfrac{1}{2} R_U \left| \tfrac{\alpha x_2 + \beta}{f^*} + 2 - \sqrt{\left(\tfrac{\alpha x_2 + \beta}{f^*}\right)^2 + 8 \tfrac{\alpha x_2 + \beta}{f^*}} \right|.$$

A short calculation shows that whenever $|\alpha x_2 + \beta - f^*| \leq \frac{1}{2} f^*$, this step size is at least $\frac{1}{4f^*} R_U |\alpha x_2 + \beta - f^*|$. Hence, at each step along the ray $\mathbf{r}(t) = \mathbf{r}_0 + t\mathbf{v}$ the error $|f(\mathbf{x}) - f^*|$ decreases by at least a constant factor of $1 - \frac{\alpha R_U v_2}{4f^*} < 1$, and therefore the sequence converges linearly to a point $\mathbf{r}(t^*)$ satisfying $f(\mathbf{r}(t^*)) = f^*$. □

Angle-valued functions. A similar analysis can be applied when $f(\mathbf{x})$ is an angle-valued function. The only difference is that the radius function R must be defined so that the balls $B_{R(\mathbf{x})}(\mathbf{x})$ never contain any singularities of f when $\mathbf{x}$ does not lie on a singularity of f, and that $R(\mathbf{x})$ must be zero when $\mathbf{x}$ does lie on a singularity.

The proof of Lemma A.2 remains essentially unchanged—the only difference is that Harnack tracing may also converge to a singular point (where $R(\mathbf{x}) = 0$). Lemma A.3 applies unchanged whenever Harnack tracing converges to a non-singular point $\mathbf{x}$, since the angle-valued function f may always be lifted to a continuous function in a neighborhood of $\mathbf{x}$. The rate at which rays converge to singular points is harder to establish, but we note that rays intersect $\mathcal{S}$ at non-singular points with probablility 1.

B SPATIAL EXTENT OF SOLID ANGLE LEVEL SETS

As in Section 4.2, let P denote a space polygon, *i.e.* a piecewise-linear curve with vertices $\mathbf{p}_1, \ldots, \mathbf{p}_k \in \mathbb{R}^3$, and let $\Omega_P(\mathbf{x})$ denote the signed solid angle function associated to P.

Theorem B.1. *Suppose that P is a connected curve which contains no self-intersections and lies on the boundary ∂C of a convex set C. Then the 2π level set of $\Omega_P(\mathbf{x})$ lies within C.*

Proof. It suffices to prove Theorem B.1 in the special case where the set C is the convex hull H of P. Let Σ denote the 2π level set of $\Omega_P(\mathbf{x})$. Note that if P is planar, then its 2π level set and its convex hull H are both equal to the planar polygon filling in P, so Σ is automatically contained in H. From now on, we may thus assume that P is nonplanar, and hence has a 3-dimensional convex hull.

To begin, we observe that Σ must be connected. Its boundary, P, is connected, so the only way for Σ to have multiple components would be to contain other components with no boundary. But any boundary-free level set of a harmonic function must stretch off to infinity, and $\lim_{\mathbf{x}\to\infty} \Omega_P(\mathbf{x}) = 0$, so no such components can exist.

Next, we will show that Σ does not intersect the boundary ∂H, meaning that the level set must be entirely contained in H or lie entirely outside of H. Let $\mathbf{x}$ be a point on ∂H, but not on the curve P itself. Since P is an intersection-free curve on the topological sphere ∂H, the Jordan curve theorem tells us that P divides ∂H into two different components. We will call the component of ∂H containing $\mathbf{x}$ the "outside" of P, and the component not containing $\mathbf{x}$ the "inside" of P. Since $\mathbf{x}$ lies on the boundary of the convex hull of P, all points of P must be located to one side of a plane at $\mathbf{x}$. Hence, if we project P onto a sphere centered at $\mathbf{x}$, then all points inside of P land in the same hemisphere. Furthermore, this projection is almost always injective[2], so the inside of P is mapped to a simple region within this hemisphere. Since P is nonplanar, it must contain at least one point strictly inside the hemisphere, so the inside of P cannot cover the whole hemisphere. Hence the unsigned area of the inside of P, which is equal to $|\Omega_P(\mathbf{x})|$, must be strictly less than 2π (the area of a hemisphere). So Σ cannot contain any points on ∂H.

It remains to show that Σ is contained within H rather than lying outside of H. To do so, we note that the same projection argument also shows that the zero level set of $\Omega_P(\mathbf{x})$ cannot intersect ∂H. Thus, the zero level set must lie outside of H, as $\lim_{\mathbf{x}\to\infty} \Omega_P(\mathbf{x}) = 0$. Since the zero level set and 2π level sets approach P from opposite sides[3], the 2π level set must lie inside of H. □

C BOUNDING SOLID ANGLE

In this appendix, we show that under mild assumptions on the polygon P, we can use a lower bound of $c(\mathbf{x}) = -4\pi$ to Harnack trace level sets of the solid angle function $\Omega_P(\mathbf{x})$. Recall that we can obtain a valid lower bound from a bound on the number of signed intersections that a line segment can have with the 2π level set of $\Omega_P(\mathbf{x})$ (Section 3.2). Theorem C.1 gives such a bound when P is a connected, intersection-free curve on the boundary of a convex set.

Theorem C.1. *Suppose that P is a connected curve which contains no self-intersections and lies on the boundary ∂C of a convex set C. Then the number of signed intersections between a generic line segment L and the 2π level set of $\Omega_P(\mathbf{x})$ is at most 1.*

Proof. Let Σ be the 2π level set of $\Omega_P(\mathbf{x})$. If P is planar, then Σ is also planar and thus intersects a generic line segment at most once.

Now, suppose that P is nonplanar. First, note that it suffices to consider only line segments contained within C: by Theorem B.1, $\Sigma \subset C$, so intersections between Σ and L cannot occur outside of C. Since Σ is a connected surface embedded in the topological ball C, with boundary contained in ∂C, it must divide C into two pieces. The number of signed intersections between L and Σ is 0 if the endpoints of L lie in the same region and ± 1 if the endpoints lie in different regions. Since these are the only possibilities, there can only be -1, 0, or 1 signed intersections between L and Σ, as desired. □

D BOUNDING THE DIPOLE POTENTIAL

In this appendix, we walk through the derivation of a lower bound on the dipole potential, a harmonic function which arises in surface reconstruction (Equation 12). While it is often difficult to find the optimal lower bound on a harmonic function, finding a valid bound is often not too hard. The techniques used here—such as combining separate bounds on the individual terms in a sum or product—are broadly applicable, and provide a starting bound which be improved with more careful analysis if necessary. For convenience, we begin by reproducing the definition of the dipole potential:

$$f(\mathbf{x}) := \sum_{i=1}^{k} a_i \frac{(\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i}{\|\mathbf{p}_i - \mathbf{x}\|^3}. \tag{21}$$

We will show that if we fix a point $\mathbf{x} \in \mathbb{R}^3$ and radius $R \in \mathbb{R}$, where $R < \|\mathbf{p}_i - \mathbf{x}\|$ for all i, then for any point $\mathbf{y} \in B_R(\mathbf{x})$ we have

$$f(\mathbf{y}) \geq \sum_{i=1}^{k} \frac{a_i \big((\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R \big)}{\big(\|\mathbf{p}_i - \mathbf{x}\| + \operatorname{sign}\big((\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R \big) R \big)^3}. \tag{22}$$

[2] in the sense that the set of lines through $\mathbf{x}$ which intersect ∂H at more than one other point has measure zero on the sphere

[3] the Biot-Savart formula for $\nabla \Omega_P(\mathbf{x})$ shows that, near the curve, $\Omega_P(\mathbf{x})$ changes at a constant rate as you rotate around P

We proceed by considering each term of the sum separately. Since the areas a_i are positive, we compute lower bounds on the fractions

$$f_i(\mathbf{y}) \coloneqq \frac{(\mathbf{p}_i - \mathbf{y}) \cdot \mathbf{n}_i}{\|\mathbf{p}_i - \mathbf{y}\|^3}. \tag{23}$$

First, we bound the numerator. Geometrically, $(\mathbf{p}_i - \mathbf{y}) \cdot \mathbf{n}_i$ measures the signed distance from $\mathbf{y}$ to a plane defined by $\mathbf{p}_i$ and $\mathbf{n}_i$. Since $\|\mathbf{y} - \mathbf{x}\| \leq R$, its value can change by at most R, and hence

$$(\mathbf{p}_i - \mathbf{y}) \cdot \mathbf{n}_i \geq (\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R. \tag{24}$$

Next, we bound the denominator. Since $\|\mathbf{y} - \mathbf{x}\| \leq R$, the triangle inequality implies that

$$\|\mathbf{p}_i - \mathbf{x}\| - R \leq \|\mathbf{p}_i - \mathbf{y}\| \leq \|\mathbf{p}_i - \mathbf{x}\| + R. \tag{25}$$

Finally, we combine the two to bound f_i. Our denominator bounds are positive, since $R < \|\mathbf{p}_i - \mathbf{x}\|$, but the numerator may be positive or negative. If the lower bound on the numator is positive, we lower bound f_i by dividing our numerator bound by the upper bound on the denominator. On the other hand, if the numerator bound is negative, we should divide our lower bound on the numerator by the *lower* bound on the denominator. So we conclude that

$$f_i(\mathbf{y}) \geq \begin{cases} \frac{(\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R}{(\|\mathbf{p}_i - \mathbf{x}\| + R)^3} & \text{if } (\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R \geq 0 \\ \frac{(\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R}{(\|\mathbf{p}_i - \mathbf{x}\| - R)^3} & \text{if } (\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i - R < 0 \end{cases} \tag{26}$$

The only difference between the cases is the sign of the R term in the denominator, so we can write the bound concisely as Equation 22.

E HARNACK TRACING IN FOUR DIMENSIONS

To run Harnack tracing in four dimensions, we take steps of size

$$\rho_{\mathbb{R}^4} \coloneqq R\left|\frac{u}{3} - \frac{a}{u} - 1\right|, \quad \text{where } u \coloneqq \left(3\sqrt{3a^3 + 81a^2} + 27a\right)^{1/3}, \quad \text{and } a \coloneqq f(\mathbf{x}_0)/f^*. \tag{27}$$

To derive this step size, we start from the Harnack inequality in 4D:

$$\frac{(R-\rho)R^2}{(R+\rho)^3} f(\mathbf{x}_0) \leq f(\mathbf{x}) \leq \frac{(R+\rho)R^2}{(R-\rho)^3} f(\mathbf{x}_0).$$

As before, we can ensure that $f(\mathbf{x})$ remains greater than a lower bound f_-, by picking a step ρ such that

$$\frac{(R-\rho)R^2}{(R+\rho)^3} f(\mathbf{x}_0) \geq f_-,$$

or equivalently, letting $a_- \coloneqq f(\mathbf{x}_0)/f_-$ and $\tilde{\rho} \coloneqq \rho/R$:

$$\tilde{\rho}^3 + 3\tilde{\rho}^2 + (a_- + 3)\tilde{\rho} + (1 - a_-) \leq 0.$$

This polynomial has a single real root (since its discriminant $-4a_-^2(27 + a_-)$ is negative) and the root is given by

$$\tilde{\rho}_{\text{lower}} = \frac{u}{3} - \frac{a_-}{u} - 1, \quad \text{where } u = \left(3\sqrt{3a_-^3 + 81a_-^2} + 27a_-\right)^{1/3}.$$

Hence, any step size below $R\tilde{\rho}_{\text{lower}}$ must be safe. Similarly $f(\mathbf{x})$ remains less f_+, so long as our step size is at most

$$\tilde{\rho}_{\text{upper}} = -\frac{u}{3} + \frac{a_+}{u} + 1, \quad \text{where } u = \left(3\sqrt{3a_+^3 + 81a_+^2} + 27a_+\right)^{1/3},$$

and $a_+ \coloneqq f(\mathbf{x}_0)/f_+$. In either case, Equation 27 provides the largest step whose safety we can guarantee using the Harnack inequality.

E.1 Bounding Harmonic Extensions

In Figure 21, we use Harnack tracing to visualize the $\frac{1}{2}$ level set of

$$f(x, y, z) = \sin(6x)\sin(6y)\sin(6z) + \sin(2x)\sin(2y)\sin(2z),$$

which is a sum of Laplacian eigenfunctions with eigenvalues -108 and -12 *resp.* We construct a harmonic extension *à la* Section 4.7.1:

$$\begin{aligned} \bar{f}(x, y, z, w) = \quad & e^{\sqrt{108}w}\sin(6x)\sin(6y)\sin(6z) \\ & + e^{\sqrt{12}w}\sin(2x)\sin(2y)\sin(2z). \end{aligned}$$

We found a very loose bound sufficed to run Harnack tracing at 120 frames per second. In particular, we use

$$c(\mathbf{x}) = -e^{6\sqrt{3}R(\mathbf{x})} - e^{2\sqrt{3}R(\mathbf{x})},$$

with a constant radius function $R(\mathbf{x}) = .15$ for simplicity. As in Appendix D, we derive this formula by bounding each term in the sum separately and summing the bounds. For any $f = \sum_i a_i\phi_i$ where ϕ_i are Laplacian eigenfunctions bounded between -1 and 1 with eigenvalues λ_i, we have a bound of $c = -\sum_i e^{R\sqrt{-\lambda_i}}$.

E.2 Stereographic Projection

In Section 4.7 we run 4D Harnack tracing on the 3-sphere $S^3 \subset \mathbb{R}^4$, mapped onto $\mathbb{R}^3$ via stereographic projection $\varphi : \mathbb{R}^3 \to \mathbb{R}^4$. Here $\varphi(\mathbf{x}) \coloneqq \left(2x_1, 2x_2, 2x_3, \|\mathbf{x}\|^2 - 1\right)/\left(1 + \|\mathbf{x}\|^2\right)$. We account for the distortion induced by φ by taking steps of size

$$\rho_{\text{stereo}} \coloneqq \tfrac{1}{2}\rho_{\mathbb{R}^4}\left(\|\mathbf{x}\|^2 - \min(\mathbf{x} \cdot \mathbf{v}, 0)^2 + 1\right), \tag{28}$$

where $\rho_{\mathbb{R}^4}$ is the step size determined by Equation 27.

Concretely, suppose that we have a ray $\mathbf{r}(t)$ in $\mathbb{R}^3$ and a harmonic function $f(\mathbf{x})$ defined on $\mathbb{R}^4$, with target level set f^*. We compute the first time t such that $f(\varphi(\mathbf{r}(t))) = f^*$. To start, we pick a radius function $R(\mathbf{x})$ and a compatible lower bound $c(\mathbf{x})$ defined on all of $\mathbb{R}^4$. Then at any time t, we evaluate $R(\varphi(\mathbf{r}(t)))$, $c(\varphi(\mathbf{r}(t)))$, and $f(\varphi(\mathbf{r}(t)))$. After shifting f and f^* by c, we find a step size $\rho_{\mathbb{R}^4}$ in $\mathbb{R}^4$ using Equation 27. We then substitute $\rho_{\mathbb{R}^4}$, $\mathbf{r}(t)$ and $\mathbf{v}$ into Equation 28 to find a step size ρ_{stereo} which is safe in $\mathbb{R}^3$, and increment t by ρ_{stereo}. As usual, we repeat until f is sufficiently close to f^*.

The inset figure in Section 4.7 shows the level sets of

$$f(x, y, z, w) = x^3y + xy^3 - 3xyw^2 - 3xyz^2,$$

using radius function $R(\mathbf{x}) = 1.25 - \|x\|_{\mathbb{R}^4}$ and bound $c(\mathbf{x}) = -1.5$.

Derivation. To derive Equation 28, we recall that φ is conformal, scaling the region around $\mathbf{p} \in \mathbb{R}^3$ by a factor of $\frac{1}{2}(\|\mathbf{p}\|^2 + 1)$ (see *e.g.* Lee [2018, Prop. 3.5]). Hence, given a curve Γ in $\mathbb{R}^3$, its image $\varphi(\Gamma)$ on S^3 has length $\int_\Gamma \frac{2}{\|\mathbf{p}\|^2+1}\, d\mathbf{p}$, which is at most $2|\Gamma|/(\min_{\mathbf{p}\in\Gamma}\|\mathbf{p}\|^2 + 1)$. So to ensure that $\varphi(\Gamma)$ stays within a ball of radius $\rho_{\mathbb{R}^4}$ in $\mathbb{R}^4$, it suffices to ensure that $|\Gamma| \leq \rho_{\mathbb{R}^4}(\min_{\mathbf{p}\in\Gamma}\|\mathbf{p}\|^2 + 1)/2$.

Now, suppose that we are taking a step from position $\mathbf{x}$ in direction $\mathbf{v}$. If $\mathbf{v}$ points away from the origin (*i.e.* $\mathbf{v} \cdot \mathbf{x} > 0$), then the minimum value of $\|\mathbf{p}\|^2$ along our step is $\|\mathbf{x}\|^2$, so we can take a step of size $\sigma \coloneqq \rho_{\mathbb{R}^4}(\|\mathbf{x}\|^2 + 1)/2$. Otherwise, the minimum value of $\|\mathbf{p}\|^2$ is—at the very least—bounded by the minimum value of $\|\mathbf{p}\|^2$ over the entire ray $\mathbf{x} + t\mathbf{v}$, which is $\|\mathbf{x}\|^2 - (\mathbf{x} \cdot \mathbf{v})^2$. Hence, the step in Equation 28 never takes us beyond the safe radius $\rho_{\mathbb{R}^4}$ in $\mathbb{R}^4$.

Received January 2024