

Uniform Sampling of Surfaces by Casting Rays

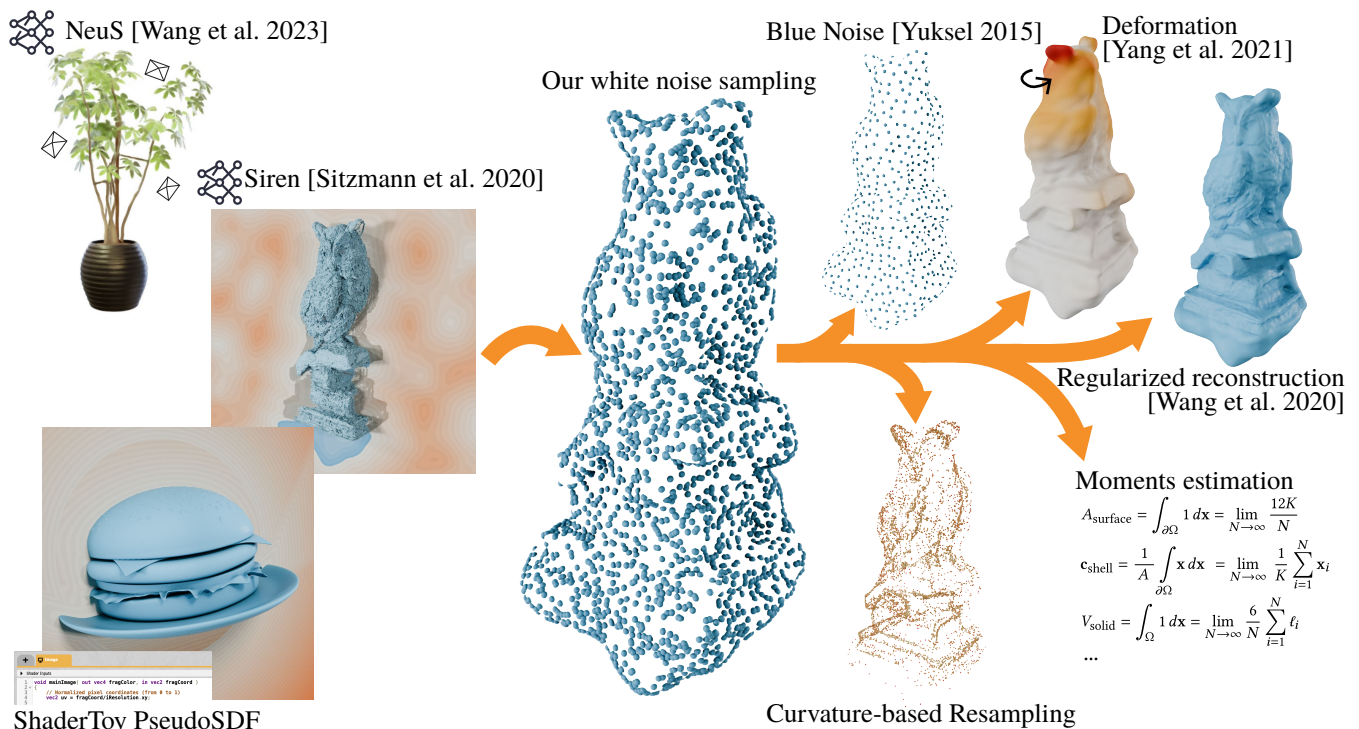
Selena Ling¹, Abhishek Madan¹, Nicholas Sharp³ and Alec Jacobson^{1,2}¹University of Toronto, Canada²Adobe Research, Canada ³NVIDIA, USA

Figure 1: Implicit surfaces represent the geometry of a shape as the zero level set of a function of 3D space. There are many varieties of implicit surfaces used in practice (left). We study a method for a fundamental surface operation: uniformly random point sampling. Any point on the continuous surface is equally likely to be chosen through our proposed process of sampling random rays in space and finding intersections with the implicit (typically with sphere tracing). This white noise sampling can then act as the raw ingredient for downstream recipes, including blue noise sampling, curvature-based importance resampling, deforming neural implicits, sampling-based regularization terms for reconstruction, and direct estimation of basic shape quantities (surface area, enclosed volume, center of mass, etc.). ShaderToy PseudoSDF credit to ©Xor (CC BY-NC-SA 3.0)

Abstract

Randomly sampling points on surfaces is an essential operation in geometry processing. This sampling is computationally straightforward on explicit meshes, but it is much more difficult on other shape representations, such as widely-used implicit surfaces. This work studies a simple and general scheme for sampling points on a surface, which is derived from a connection to the intersections of random rays with the surface. Concretely, given a subroutine to cast a ray against a surface and find all intersections, we can use that subroutine to uniformly sample white noise points on the surface. This approach is particularly effective in the context of implicit signed distance functions, where sphere marching allows us to efficiently cast rays and sample points, without needing to extract an intermediate mesh. We analyze the basic method to show that it guarantees uniformity, and find experimentally that it is significantly more efficient than alternative strategies on a variety of representations. Furthermore, we show extensions to blue noise sampling and stratified sampling, and applications to deform neural implicit surfaces as well as moment estimation.

1. Introduction

Sampling uniformly distributed points on surfaces is essential for characterizing the underlying geometry in many downstream applications such as visualization and simulation. Concrete algorithmic use-cases include integrating surface areas or other geometric properties [LWMB03], evaluating regularizers for geometric optimization [YWOSH21], computing metrics such as Chamfer distance [HNSS*24], and sampling BSSRDF exit points in rendering [KKCF13]. It is easy to draw samples on a surface explicitly defined as a mesh, however it is not so straightforward when no mesh is available, such as implicit surfaces. This is problematic, as implicit surfaces are increasingly widely used, for instance as a neural representation for shapes in machine learning. We consider the problem of sampling uniformly-distributed points on surfaces in this more general setting.

How might one sample points on an implicit surface? One possibility is rejection sampling, drawing random points in space and keeping only those near the surface, but this wastes many samples and requires an error-inducing projection onto the surface. Another is to use isosurface extraction such as marching cubes to recover a mesh, and sample from that, but this requires expensive sampling to a grid and can alias fine features. Principled sampling processes like Markov chain Monte Carlo guarantee a proper distribution in an asymptotic limit, but convergence may be slow in practice.

Instead, we leverage a classic mathematical relationship to the intersection of random lines with the surface (e.g., “Cauchy-Crofton Formula” and “Buffon’s Needle Problem”). Taking random lines drawn from an appropriate distribution and gathering all intersections yields a uniform sampling of the surface—see [PPK16] for one reference. Precisely, this strategy produces a white-noise uniform distribution of samples, although other samplings can also be obtained with our method (Figure 2). This relationship is well-known in mathematics. It has sporadically and briefly been studied in visual computing [DHJM08], but has not previously been put to work in the context of modern implicit surfaces and neural representations, where we show that it offers significant benefits.

Importantly, this sampling method applies to any surface representation for which we have the ability to intersect rays with the surface. Because ray casting is already a necessary operation for rendering and visualization, it is widely and efficiently implemented for a wide variety of implicit surfaces and other non-standard shape representations. For instance, we can efficiently intersect rays with implicit signed distance functions surfaces via sphere tracing [Har96], with harmonic functions via Harnack tracing [GYBC24], or more general implicit functions via interval tracing [SDF97]. Even recent Gaussian particle representations allow for efficient ray tracing [MLMP*24].

In this paper we study ray intersection-based sampling in the context of implicit surfaces and neural representations. We provide self-contained proofs supporting the claim of uniformity and numerically compare with other existing approaches, showing significantly improved efficiency and uniformity. We also demonstrate extensions to downstream applications of the approach, including stratified sampling (Section 4.3), blue noise sampling (Section 6.4), implicit deformation (Section 6.5), and more.

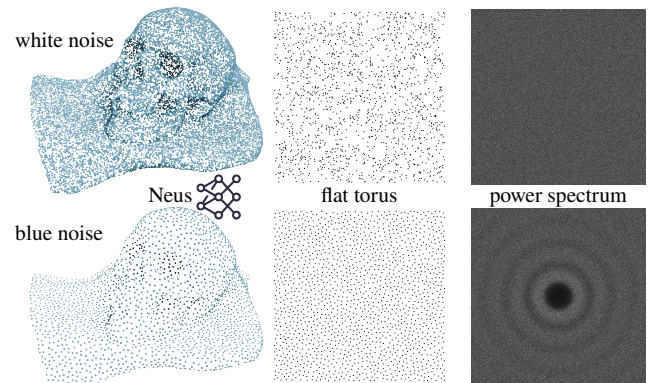


Figure 2: White noise sampling is often useful as a raw ingredient to downstream applications, such as blue noise generators (e.g., [Yuk15, Bri07]). Our white noise samples on a neural implicit [WLL*21] (top left) are subsampled to a blue noise sample set [Yuk15]. For reference, we show the same process on a periodic square and its corresponding power spectrum images [SD11].

2. Related Work

Sampling surfaces is a core operation used across visual computing from rendering to numerical integration and beyond. Here we focus specifically on past work for sampling when explicit mesh representations are not available, such as implicit surfaces.

2.1. Sampling and Implicit Surfaces

Implicit surfaces are a flexible and general representation, defining surfaces as the level set of functions such as *signed distance functions* or *occupancy functions*. Recent research across visual computing has leveraged neural implicit functions as a shape representation for tasks in learning and reconstruction [PFS*19, WLL*21, YGKL21]. Across these works, sampling the level set has proven to be helpful for many downstream tasks such as rendering from the underlying shape [YGKL21, WLL*21], improving its optimization by defining on-surface regularization terms [AHY*19, YWOSH21, YGKL21, HNSS*24, HCHH24] and shape manipulation [YBHK21].

2.2. Sampling Algorithms

Although it is useful to sample points from surfaces which lack an explicit representation, it is not obvious how to do so. Many strategies have been considered in past work.

Rejection Sampling A basic standard approach is *rejection sampling*: drawing points at random in the domain and keeping only those which lie on the surface, see e.g. [YBHK21, HNSS*24]. Rejection sampling is simple and straightforward, but may require a huge number of rejected samples. Furthermore, because surfaces are co-dimensional sheets with no volume, points will never land exactly on the surface—in practice one must either retain samples

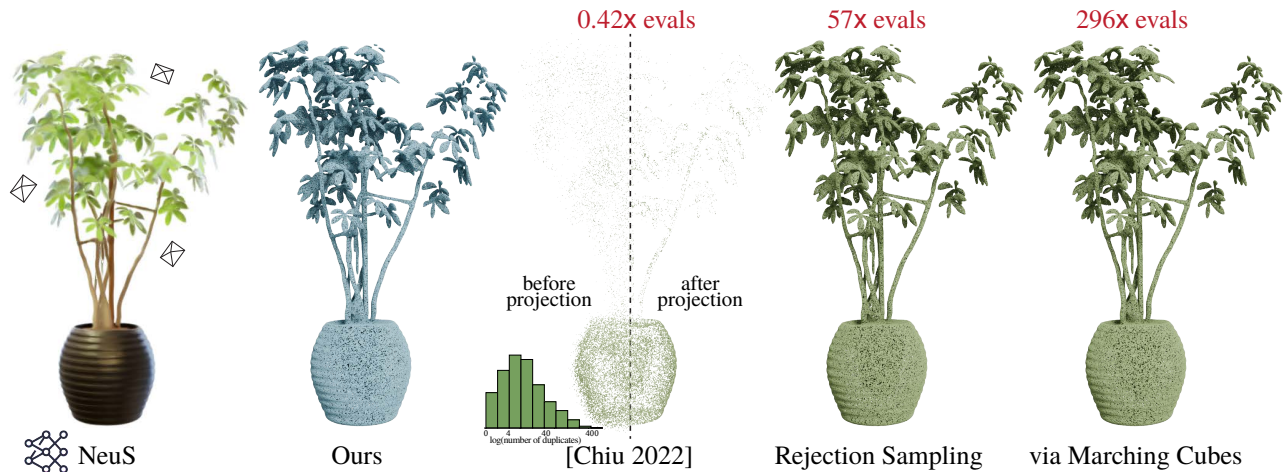


Figure 3: We visualize 500,000 samples using our method and all baseline methods on the level set of a neural implicit function [WLL*21] learned from images of the ficus scene in the Blender dataset. Our method achieves similar or better sample quality as rejection sampling and sampling via marching cubes on a grid of resolution 4096^3 while being a factor of $57\times$ and $296\times$ cheaper in terms of function evaluations, respectively (for reference, ours took 2.3×10^8 function evaluations). Because the specialized Hamiltonian Monte Carlo method from [Chi22] can only sample near the surface (left slice), it requires an additional non-trivial projection to result in on-surface samples (right slice). This method also results in duplicated samples (which are necessary for correct statistics), so we also plot a histogram of duplicated sample counts. Both rejection sampling and sampling via marching cubes also require projection, but is too small to visualize (though it does mildly affect exact uniformity).

in a narrow band around the surface (not truly sampling the surface), or project onto the surface as a post-process (concentrating points in positively-curved regions).

Grid Based Another possibility is to evaluate an implicit function on a regular grid for sampling. Most commonly, isosurfacing algorithms such as marching cubes [LC98] are used to construct an explicit mesh as an intermediary for sampling [DBD*22, HNSS*24, HCHH24]. Yan et al. [YWW14] apply a related grid-decomposed sampling by observing that the surface can be represented as a height function per-cell, although the function is still linearized within each cell. These strategies nicely leverage the well-established tools of extraction and mesh sampling, but may suffer from excessive computation and aliasing of fine features from working on a grid.

Particle Evolution and MCMC Other methods iteratively update a sample set to converge to the desired distribution to gradually discover a suitable set of samples. Wang et al. [YWOSH21] approximate blue noise samples to assist in training, via a three-stage procedure of projecting, resampling, and upsampling, although they target improved training efficacy more so than any precise sampling distribution. Such strategies can be formalized via Markov chain Monte Carlo (MCMC), which provably converge to a uniform sample set in the asymptotic limit, as studied in the thesis of Chiu [Chi22] for implicit surface sampling. Likewise, Langevin dynamics-based formulations evolve to the desired distribution according to a stochastic differential equation [CYAE*20, YBHK21], and [LLK*22] sample from a mollified interaction energy. How-

ever, these methods may require large numbers of iterations to “warm up” to a uniform distribution on complex geometry, and most still require an error-inducing projection step to generate points exactly on the surface.

Line Intersections The sampling method studied in this work follows from a classic relationship of random lines intersecting a surface, see Section 3. Although this approach has not yet been utilized in recently important applications with implicit surfaces where it has significant advantages, it has occasionally appeared elsewhere in visual computing, which we outline here. Detwiler et al. use random lines to sample points specifically in the context of CAD geometry [DHJM08], although their approach requires specifying an upper bound on the number of intersections *a priori*, which may be prohibitive in practice. The same approach is adapted to point clouds [LYZ*06] via the ray tracing approach of [SJ00], and digital binary-voxel geometry [LYZ*10], both for the particular purpose of estimating surface area. Some other works use low-discrepancy sequences to obtain surface samples on CAD objects [LWMB03] for estimating surface area, and for sampling on meshes [RWCS05]. Beyond surface sampling, ray marching has also been used for sampling on the medial axis [YDL*14].

2.3. Other Notions of Sampling

We primarily consider uniform sampling in the sense of white noise, but other kinds of sampling are also important in visual computing. Low discrepancy sequences and stratified sampling [LWMB03, RWCS05, Qui09] seek to reduce variance at

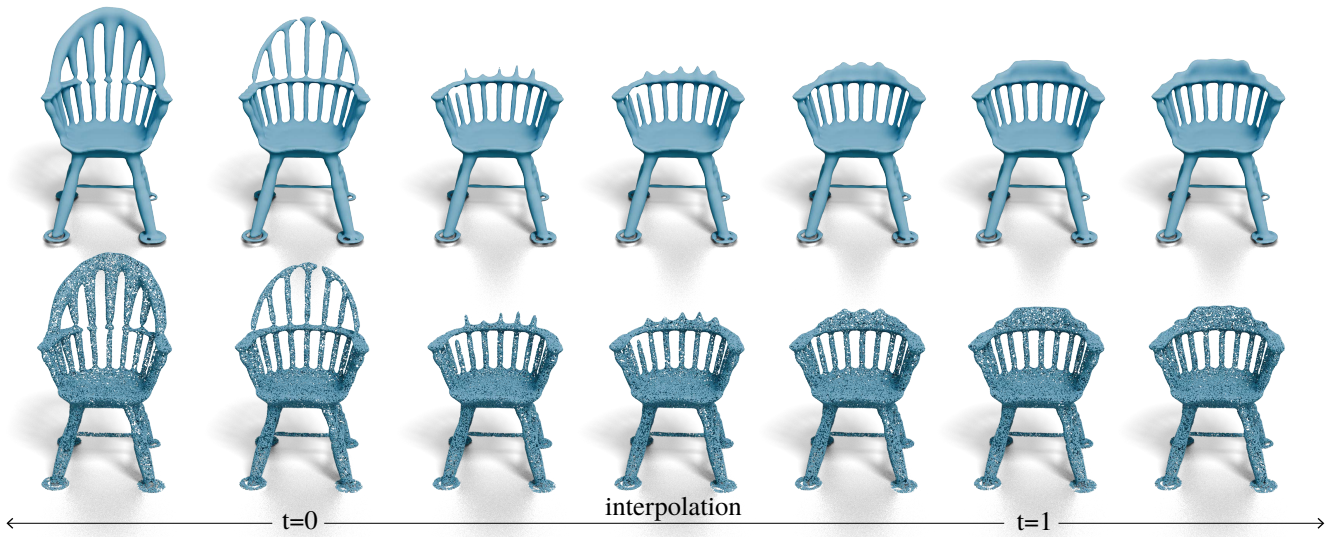


Figure 4: A neural field is learned to represent a smooth interpolation between two different chairs at $t = 0$ and $t = 1$ as in [LWJ*22] (top). Our method uniformly samples the implicit surface of the interpolated shape at any point in the sequence (bottom).

bounded sample counts by spatially distributing the sample sequence across the domain. Blue noise sampling seeks a different distribution, with nicely-spaced samples on the surface which are useful for texture synthesis and perceptual optimization [Bri07, Yuk15, ÖAG10]; in Section 6.4 we show how our sampler can do the same on surfaces.

3. Theory

Our method is derived from the relationship between random lines and their intersections with a surface. These properties are well-known in integral geometry—the Cauchy-Crofton Formula relates the length of a curve to the number of intersections with a random line, while Buffon’s needle problem considers the likelihood of a line segment intersecting parallel strips. One introduction to sampling via line intersections can be found in [PPK16]. In this section, we provide a self-contained introduction to equidistributed sequences and some intuition for why the algorithm works.

3.1. Equidistributed Sequences

A crucial building block of randomized algorithms is an underlying sequence of random numbers that are uniformly distributed, as they provide a simple and flexible primitive for stochastic computation. Instead of attempting to obtain truly random numbers, pseudo-random number generators (PRNGs) are used in practice, which produce long deterministic sequences with statistical properties that emulate truly random uniformly distributed sequences. One desirable property is that the proportion of samples in a given region of the sampling domain should be roughly equal to the relative area of that region; sequences that exhibit this property are called *1-equidistributed* [Fra63]. An equivalent characterization is that, for any function f over the sam-

pling domain Ω (an m -dimensional manifold embedded in \mathbb{R}^n for any $1 \leq m \leq n$), the sequence $\{x_i\}$ can be used to construct a Monte Carlo estimator of $\int_{\Omega} f(x) dx$ with uniform contribution weights that converges as the sample count N approaches infinity (i.e., $\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^N |f(x_i)| = \int_{\Omega} f(x) dx$) — such an estimator is called *consistent*. The notion of equidistribution can be extended to arbitrarily large k by instead considering the convergence of the estimator $\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{i=1}^{N-k+1} |f_k(x_i, \dots, x_{i+k-1})| = \int_{\Omega^k} f_k(x_1, \dots, x_k) dx_1 \dots dx_k$ for a function of k variables f_k . If a sequence is k -equidistributed for all natural numbers k , it is *completely equidistributed*. Since k -equidistribution for increasingly large k imposes more and more uniformity requirements on the sequence, it suffices to use k -equidistributed sequences for large k in numerical computation, in place of samples from a truly uniform distribution.

3.2. Sampling via Oriented Line Casting

The surface sampling algorithm is as follows: uniformly sample M oriented lines, and for each line, append *all* of its surface intersections with Ω into a list of points (this algorithm was previously proposed by Palais et al. [PPK16]). At first glance, it is surprising that such a method truly produces uniform surface samples, but building upon the Cauchy-Crofton formula from integral geometry [San76], by taking all surface intersections for each line rather than, e.g., a single intersection per line, the resulting sequence is k -equidistributed if the line distribution is also k -equidistributed.

Below we provide some intuition for the two key ideas behind the proof for the 1-equidistributed case; the general k -equidistributed case follows similarly (and see [PPK16] for a rigorous proof). The first key idea is that each point on the surface is intersected by exactly one line along each direction $\vec{d} \in S^2$, so as long

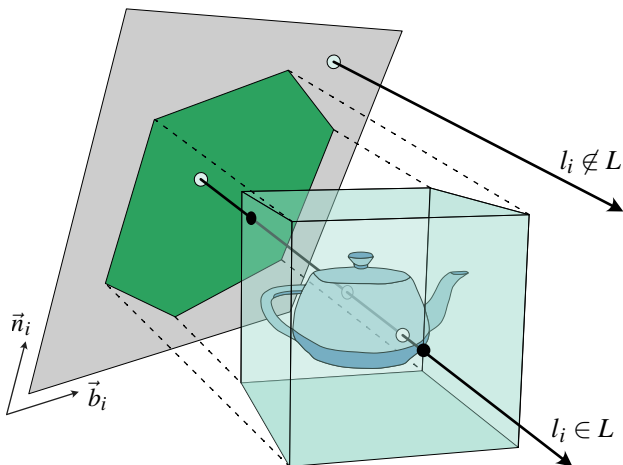


Figure 5: An illustration of our uniform ray sampling algorithm as in Algorithm 1. Given the blue $[-1, 1]^3$ cube around an implicitly-defined teapot shape, we first sample a direction \vec{d}_i uniformly. Among all rays parallel to that direction, we only want the rays that intersect the cube to add to our ray sample set L , which we achieve by rejection sampling on the gray $2\sqrt{3}$ -sided square in the plane spanned by normal \vec{n}_i and bi-normal \vec{b}_i . The points whose emanated rays intersect the bounding box fall in the green region on the plane, which is the orthographically projected area of the bounding box onto the plane.

as we uniformly sample lines in every direction within a region enclosing the surface, we can “average out” the contribution of a point to an arbitrary integral over the surface over all line directions. The second key idea is that different lines can intersect different numbers of points, which means the lines need non-uniform densities (e.g., lines with more intersections should have a larger density). Although it is intractable to obtain the true line densities *a priori*, returning every intersection along the line essentially amortizes a line’s true non-uniform density and gives each point the desired uniform density, and so the consistent Monte Carlo estimator can simply use uniform weights. An alternative strategy was used by Detwiler et al. [DHJM08] where they rejected lines based on the number of intersections relative to the maximum possible number of intersections with Ω , but in our approach extracts at least one surface sample from every ray and can extract several samples for rays with many intersections, making it more efficient while still maintaining uniformity.

The samples produced by this method are not independent (though they are identically distributed), as the collinear points produced by intersections with the same line are correlated. However, the definition of k -equidistribution ensures that the points are sufficiently well-distributed such that the correlation between samples does not influence downstream applications.

For the rest of the paper, we will refer to *rays* instead of oriented lines; although rays have an associated origin and lines do not, as we will describe in Section 4.1, rays are sampled by selecting an

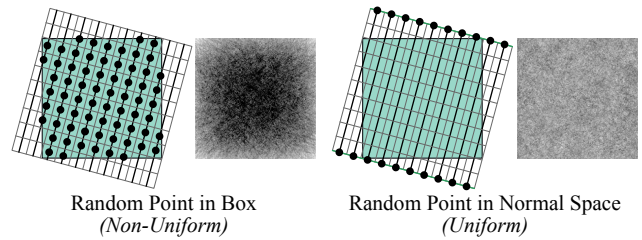


Figure 6: Ray origin sampling is crucial for obtaining uniform ray samples. If origins are simply chosen as random points within the box (left), the rays will pass through the center of the box more frequently than the edges (middle left). Meanwhile, if the origins are chosen in the space normal to the line direction (middle right), then the ray density is uniform over the box (right).

origin outside of the bounding cube, such that exactly one ray corresponds to each oriented line (see Figures 5, 6).

3.3. Ray Resampling

The one-to-many sampling procedure described above can also be viewed through the lens of resampling [Rub87, Tal05]: each ray can be given a weight proportional to the number of their intersections with the surface, and then the rays can be resampled with replacement with probabilities to their weights, after which an intersection along the ray is randomly selected with equal probability. Taking every single intersection for each ray is then a particular instantiation of the aforementioned resampling process, where each ray ℓ_i with k_i intersections is resampled k_i times, and each time a different intersection along the ray is selected. We formally prove the equivalence that resampling produces uniformly distributed samples in Appendix A, and empirically show that resampling produces samples of comparable quality in Figure 7.

4. Method

In this paper, we propose a method that uniformly samples the level set of a given implicit function via random ray casting and finding the intersections between the rays and the level set within the bounding domain. The inputs to our algorithm is a bounding box defining the sampling domain and an implicit function with a Lipschitz constant bound λ . The samples extracted from our method lies strictly on the implicitly-defined surface without needing additional projection and is uniformly distributed on the surface by the definition of uniformity we provided in above section.

4.1. Uniform Ray Sampling

Sampling rays uniformly without any bias is an essential first step in our method. However, it is impossible to sample directly in an unbounded manner in the same way that sampling all points in \mathbb{R}^3 is impractical. Our method is only interested in rays that will possibly intersect the level set defining the surface, and therefore it is reasonable to restrict the problem of uniform ray sampling to uniformly

Algorithm 1 UniformRays: Sampling n uniformly distributed rays within bounding box $[-1, 1]^3$.

Require: M : number of rays

Require: $bbox$: bounding box of size $[-1, 1]^3$

```

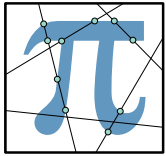
 $L \leftarrow \{\}$ 
while  $size(L) < M$  do
   $\vec{d}_i \leftarrow$  random unit direction
   $\vec{n}_i, \vec{b}_i \leftarrow$  normal, bi-normal direction of  $\vec{d}_i$  via SVD
   $u_0, u_1 \leftarrow$  random offset values uniformly sampled in the interval between 0 and the bounding box diagonal length  $[-\sqrt{3}, \sqrt{3}]$ 
   $o_i \leftarrow u_0 \vec{n}_i + u_1 \vec{b}_i$ 
   $\ell_i = o_i + t \vec{d}_i$ 
  if  $IntersectBoundingBox(\ell_i, bbox)$  then
    Append  $\ell_i$  to  $L$ 
return  $L = \{\ell_i\}$ 

```

sampling all rays that intersect the bounding box of the surface. Note that although many other alternative bounding volumes exist, e.g., a bounding sphere, we default to a non-tight bounding box following the convention in deep learning of normalizing shapes into the $[-1, 1]^3$ cube, though tight bounding volumes would reduce the number of sampled rays that miss the surface and would thus be more efficient. While there are a variety of constructions, we propose to first sample a direction uniformly, and then among all rays parallel to that direction, uniformly sample ray origins so the resulting ray intersects the bounding cube. We do this using rejection sampling, by uniformly sampling a point on a $2\sqrt{3}$ -sided square in the plane defined by the normal and bi-normal directions \vec{n}_i and \vec{b}_i , respectively, until the parallel ray passing through this point intersects the bounding box. This intersection test between a ray candidate and the bounding box can be done with a fast ray-slab test, which has a low rejection rate if the bounding box is tight. See Algorithm 1 for a more detailed version of this algorithm and Figure 5 for an explanatory diagram. This algorithm generalizes to any bounding primitive (e.g., spheres, k-DOPs), but bounding spheres can skip the rejection sampling step and instead directly sample ray origins on the projected disc on the (\vec{n}_i, \vec{b}_i) plane.

One might wonder if a simple ray sampling algorithm would suffice, for example, picking a random point in $[-1, 1]^3$ and pairing it with a random direction. However, this results in a non-uniform distribution as shown in Figure 6.

4.2. Ray-Intersection Evaluation



Once we have M randomly sampled rays, we want to find all intersections between the sampled rays and the surface (see inset, green points). For implicit functions, we consider a known Lipschitz bound of λ , i.e.,

$$|f(p_1) - f(p_2)| \leq \lambda \|p_1 - p_2\|_2, \lambda > 0.$$

This means that at every query location x , we know we can safely take a step of size $|f(x)|/\lambda$ in any direction \vec{d} without stepping over the zero level set [Har96]. We slightly modify the classic sphere tracing algorithm [Har96], originally designed to find only the first

Algorithm 2 UniformPoints: Sample uniformly distributed points on the zero level set of an implicit function f with Lipschitz bound λ via random ray casting.

Require: f : implicit function with Lipschitz bound λ

Require: $\epsilon = 1e^{-4}$: tolerance for ray-intersection finding

$L \leftarrow$ Sample M rays via UniformRays() in Algorithm 1

for $i = 1, 2 \dots M$ **do**

$P = \{\}$

$t = 0$

$p \leftarrow o_i + t \vec{d}_i / \lambda$

$s \leftarrow |f(p)|$

while $t < length(\ell_i)$ **do**

if $s < \epsilon$ **then**

Append p to P

while $s < \epsilon$ **do**

$t \leftarrow t + \max(s, \epsilon)$

$p \leftarrow o_i + t \vec{d}_i / \lambda$

$s \leftarrow |f(p)|$

$t \leftarrow t + s$

$p \leftarrow o_i + t \vec{d}_i / \lambda$

$s \leftarrow |f(p)|$

return P

intersection of a given ray with the level set, to instead find all intersections of a ray with the level set. In our modification, we ray march from the origins of each ray with step size $|f(x)|/\lambda$ until we find an intersection at the zero level set when $|f(p)| < \epsilon$. After each intersection, we keep ray marching with step size $\max(|f(x)|, \epsilon)/\lambda$ until the implicit function value reaches above the threshold value ϵ , and proceed as usual to find the next intersection, terminating when we reach the end of the ray (i.e., the ray is outside of the box). Please see Algorithm 2 for the detailed pseudocode.

As this part of our algorithm is largely based on sphere tracing, our method can benefit from any improvements proposed for sphere tracing such as [GGPP20].

Taking One Intersection A tempting alternative is to simply take one random intersection instead of all intersections on each casted ray. However, as shown in Figure 7, naively only keeping one sample on casted rays results in incorrect non-uniform samples.

4.3. Acceleration and Stratification via Sparse Voxels

Many neural implicit surface functions come with existing sparse voxel structures built to accelerate its optimization [MESK22]. Our method naturally benefits from sparse voxel structures as we can divide every implicitly-defined surface into sub-surfaces with open boundaries inside each voxel, and apply our method independently within each voxel. Since our method only requires an implicit function of non-zero Lipschitz bound, it can work with shapes with non-manifold junctures and open boundaries as shown in Figure 8.

Applying our method with sparse voxel structures not only accelerates the sampling process but also provides a way to perform stratified sampling to reduce variance in Monte Carlo estimates, as shown in Figure 9. The sparse voxel structure is also useful for input where sphere tracing-like algorithms cannot be efficiently used

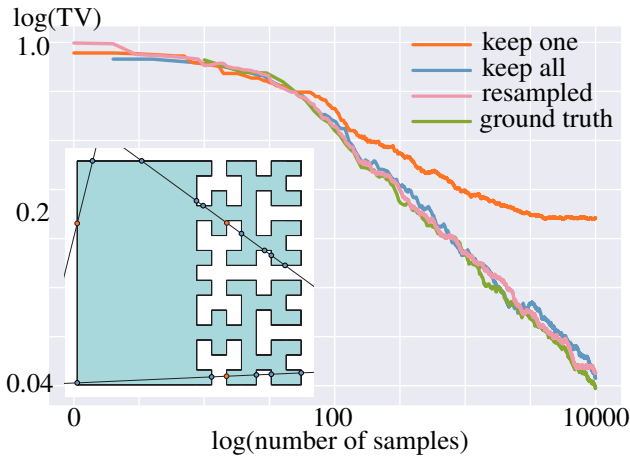


Figure 7: We plot the total variation (TV) score measuring the uniformity of sample sets acquired with uniform sampling on the ground truth polyline (“ground truth”), our method (“keep all”), the resampling procedure described in Section 3.3 (“resampled”) and taking a single intersection (“keep one”). Our method and the resampling approach produce uniform samples on the surface comparable with comparable TV scores to directly sampling on the polyline, while the keeping one sample along each casted ray produces worse TV scores.



Figure 8: Our method only requires the surface to define a codimension-one shape as the level set of an implicit function. Non-manifold junctures and open boundaries are no issue for our white noise sampling, such as this sailboat modeled as a cascade of analytic unsigned distance functions.

everywhere, e.g., many formulations of neural implicits which encourage SDF-ness but do not guarantee it and may have large global Lipschitz bounds.

5. Evaluations

An ideal uniformly distributed set of samples on the surface means the ratio of samples belonging to each local surface region should be the same as the area ratio of the local region. Therefore, if we can divide the surface into disjoint local patches, one can measure the

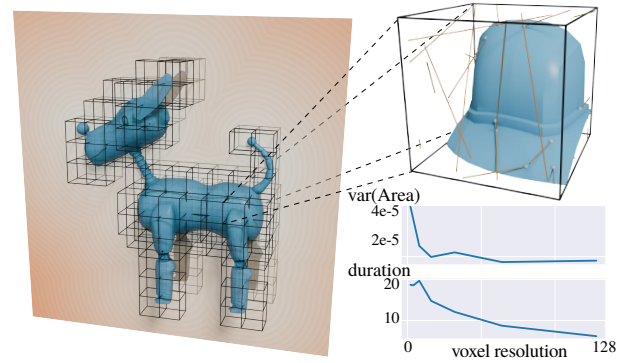


Figure 9: Sparse voxel structures can accelerate our method and act as a form of stratified sampling which reduces variance for estimations of shape quantities as described in Section 6.2. Here we show one instance of such a voxel structure with a voxel grid resolution of 16, along with a zoom-in view of one voxel, a set of sampled rays (orange) in this voxel, and the resultant samples on the zero level set (white). We plot the variance of a surface area estimate over 30 runs using sparse voxel structures built with different grid resolutions, which demonstrates the variance-reducing benefits of stratified sampling through voxels. We also plot the total time needed (in seconds) for our method across voxel resolutions using the same number of rays and show its acceleration benefits.

uniformity of a sample set by computing statistical distance metrics, such as total variation (TV) distance and KL divergence, between the distribution of samples per patch and the area density of these patches.

While in practice our method is not necessary for triangle meshes, they nevertheless provide an ideal real-world test set for measuring the behavior of our method and baselines against a ground truth sampler. We can easily produce a “ground truth” uniform distribution over the mesh by sampling triangles proportional to area and uniformly sampling on each selected triangle, and we can also evaluate our method on triangle meshes through direct ray-intersection queries or sphere tracing the minimum distance to the mesh (in our experiments, we use the latter for the sake of generality). Therefore, we focus on evaluating our method and baseline methods on signed implicit functions defined by an existing mesh dataset [MPZ14].

The surface defined by a triangular mesh has a natural disjoint decomposition into triangles, so for a given sample set on a mesh, we can measure its uniformity via the discrete total variation (TV) score by taking the sum over all triangles of the absolute difference between the proportion of samples in the triangle and the area-proportional density of the triangle:

$$\text{TV} = \frac{1}{2} \sum_i \left| \frac{n_i}{N} - \frac{A_i}{A_{\text{total}}} \right| \quad (1)$$

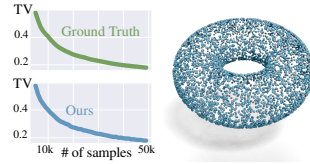
Lower TV scores indicate more uniform sample sets.

In addition to uniformity of samples, we measure the efficiency of each sampling method by counting the number of implicit func-

tion evaluation calls used during sampling. The reason we use the number of evaluations instead of run time is because many of the methods studied here can be dramatically optimized for run time due to their parallelizable nature, such as marching cubes and even our method. We believe the number of function evaluations is an accurate measurement of the method’s efficiency that is agnostic to the chosen implementation, so our research prototype still provides meaningful performance statistics.

5.1. Warm Up: Analytic Torus

To start, we apply our method to an analytic implicit distance function of a torus, and compute the TV score of the generated samples. We can sample the torus uniformly via an analytical formula derived with the inverse CDF method as the ground truth.



To evaluate uniformity of our samples against the analytical samples, we parameterized the torus into a toroidal grid of resolution 100×100 . Each grid patch’s surface area can be analytically computed via integration. We then compute the TV distance between the surface area distribution of the toroidal patches and the distribution of number of samples per patch. See the figure on the right for a plot of TV distance against the number of samples, from both our method and the analytically-derived formula. The uniformity of samples acquired from our method on the torus’s surface aligns well with the ground truth uniform sampler, verifying the theoretical uniformity of our method.

Baselines We next compare our methods to three baseline methods on a dataset of implicit functions defined by 114 meshes [MPZ14] for which we have access to the ground truth implicit function and a ground truth uniform sampler, i.e., uniformly sampling on the mesh. For baselines, we compare with (1) uniform sampling on the extracted mesh via marching cubes, (2) sampling using a Hamiltonian Monte Carlo algorithm [Chi22], and (3) rejection sampling as used in many existing neural implicit techniques [YBHK21, HNSS*24].

More specifically, baseline (2) is a Markov Chain Monte Carlo method that generates a sequence of samples following a specialized density distribution proposed in the thesis of Chiu [Chi22], which asymptotically converges to a uniform set of samples on the surface. Implementation-wise, we mostly follow the experiment setup specified in [Chi22] except we replace the initial 1000 gradient descent steps with 5 Newton descent steps before the burn-in period. Given an initial sample x randomly sampled in the bounding box and the signed distance function f , a Newton step is:

$$x = x - f(x) \frac{\nabla f(x)}{\|\nabla f(x)\|^2}.$$

This effectively moves point x to be near the surface with fewer steps than gradient descent. The rest of the hyperparameter choices stay the same: mass size is 1, total integration time is 1, the number of integration steps is 100, and the number of burn-in Hamiltonian Monte Carlo steps is 500. Note that this specialized Hamiltonian Monte Carlo method practically can only sample near the surface

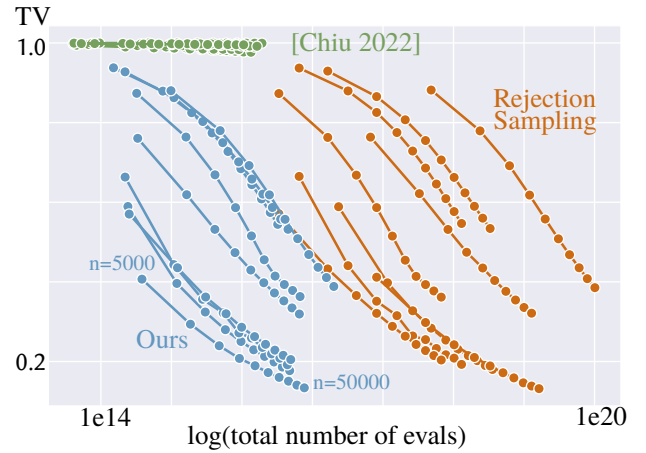


Figure 10: We compare our method with the specialized Hamiltonian Monte Carlo sampling algorithm proposed in [Chi22] and rejection sampling, by drawing an increasing number of samples from 5,000 to 50,000 on a subset of 8 different implicit functions defined by meshes from a dataset [MPZ14]. For each sampling run, we measure the uniformity of the resulted sample set via TV distance and the total number of function evaluations needed. Our method consistently outperforms both baseline methods in both TV distance and function evaluation count, resulting in more uniform sets of samples on surface while being less costly to evaluate.

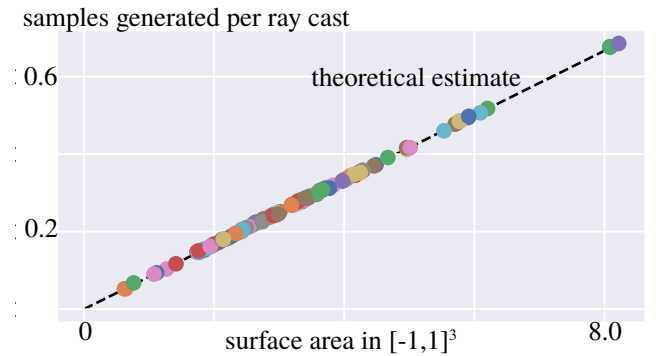


Figure 11: We ran our method on 114 different implicit surface functions from a dataset of meshes [MPZ14]. Here we plot the relationship between ground truth surface area and the ratio between number of samples and the number of casted rays (i.e., average intersection count). We confirm that the average number of samples generated per ray is linear in surface area, which corroborates the surface area formula in Section 6.2. This also means the rejection rate and wasted evaluation cost from our method correlates with the surface area, which we show in Appendix B.

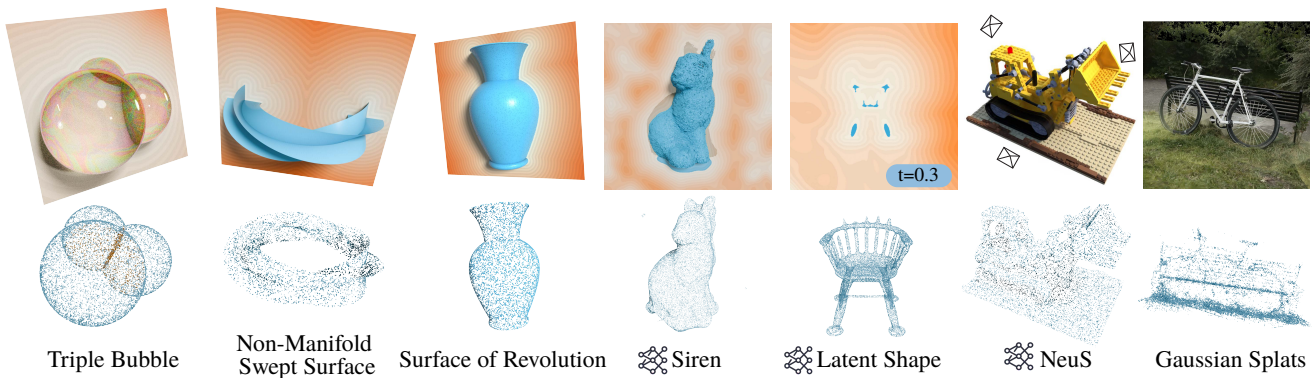


Figure 12: We apply our method to a variety of implicit functions. From left to right: a Möbius transformation of the “standard triple bubble” from [Sul12], a swept unsigned distance function resulting in a non-manifold shape, a surface of revolution generated by a cubic Bézier curve, another NeuS surface trained from the Lego scene in the Blender dataset [WLL*21], another SIREN surface fitted on LiDAR point clouds of scan55 from the DTU dataset [SMB*20], an implicit surface from interpolating two surfaces using latent code $t = 0.3$ [LWJ*22], and a preliminary result on a Gaussian Splats scene.

Table 1: We compute the average total number of function evaluations to sample 50,000 points on a dataset of surfaces [MPZ14], as well as the total variation (TV) score measuring the uniformity of the samples. Although our method is generally aimed at implicit surfaces, we use meshes here for the sake of known geometry to evaluate against. For marching cubes, the grid size is 1024^3 . The total variation “Ground truth” refers to uniform sampling using the ground truth mesh, for which we use the average score across 10 sampling runs for each shape.

Method	Ground Truth	[Chi22]	Rejection Sampling	via Marching Cubes	Ours
Number of Evals	N/A	9.98×10^6	3.98×10^8	1.07×10^9	1.92×10^7
TV	0.373	0.989	0.373	0.383	0.372

with a small but non-zero T parameter and therefore requires an additional projection step. Note also that both baselines of rejection sampling and sampling via marching cubes can produce samples far from the surface as well, depending on the chosen hyperparameters. We use the same Newton method for any projection step needed across all methods.

Results Our method achieves the best trade-off along the axes of sample uniformity and method efficiency compared to all baselines as shown in Table 1.

In Figure 10, we compare with Hamiltonian Monte Carlo [Chi22] and rejection sampling, which is a common method used in prior work for uniformly sampling implicit surfaces [YBHK21, HNSS*24]. Here we sample an increasing number of samples from 20 to 50,000 on all shapes using all methods, and plot the TV distance scores and the total number of function evaluations on a randomly-chosen subset of 8 shapes. Our method is at the optimal front with respect to both metrics compared to the other baselines, i.e. consistently resulting in more uniform samples while being less expensive. As mentioned before, the theory behind the Hamiltonian Monte Carlo algorithm only

guarantees uniform on-surface samples asymptotically when the temperature parameter T goes to 0 [Chi22], which is practically impossible. With a non-zero T , the samples usually end up floating near the surface and require an additional projection step to exactly lie on the surface. We show the samples before and after projection in Figure 3. In addition, the sequential sample sets acquired from such a Markov Chain Monte Carlo method often contain duplicated samples, resulting in a much worse uniformity measure.

We compare to sampling via marching cubes in a separate figure as the cost of function evaluation with marching cubes is tied with the grid resolution instead of the number of samples. As shown in Figure 13, we obtain 50,000 samples on one shape using our method and sampling on marching cubes-extracted meshes from grids of increasing resolution from 300^3 to 1024^3 , and plot the total variation score using the ground truth mesh. Marching cubes is consistently more expensive than ours with respect to total number of queries as it grows cubically with grid resolution, and the samples are less uniform as the mesh extracted with marching cubes tend to miss thinner structures and details. While we focus on a single example in this figure, the observations generalize as demonstrated by metrics in Table 1 computed from a dataset of shapes [MPZ14].

In addition to comparing with baselines, we provide additional analysis of our method itself. We focus on the relationship between the ground truth surface area and average number of samples per casted ray. As shown in Figure 11, we observe a strong linear correlation between the average number of samples per casted ray and the surface area of the underlying implicit surface, which is what we expect from integral geometry [San76]. This means the rejection rate and the amount of wasted computation from our method is closely tied with the underlying surface area as well, which we show in Appendix B. At the same time, this relationship allows one to use our method to estimate important shape quantities such as surface area and volume (Section 6.2).

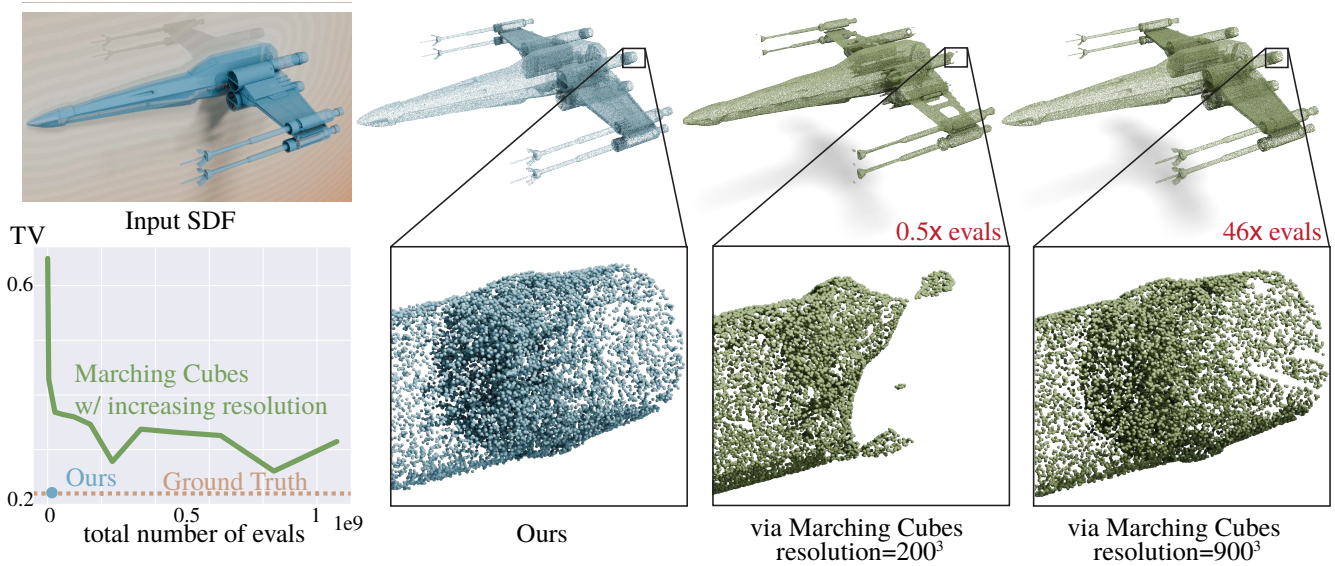


Figure 13: We compare our method with sampling a mesh extracted via marching cubes, from grids with resolutions ranging from 200^3 to 1024^3 . With this particular shape, even though the uniformity of the sample set via marching cubes improves with increasing grid resolution, the extracted surface still misses some parts of the surface with a relatively high resolution grid as shown in the zoom-in view on the right, which degrades uniformity. For sampling the same number of points, our method results in a sample set that is more uniformly distributed on the surface, measured by a lower total variation score (TV), while being much cheaper to evaluate (for reference, ours took 1.6×10^7 function evaluations). We also plot the “Ground truth” total variation score, averaged over 10 sampling runs using the ground truth mesh, for reference.

6. Applications

This section focuses on various applications of our method. Our method can be used for a range of tasks including shape quantity estimation, offset surface sampling, and curvature-based resampling. Meanwhile, extracting a uniformly distributed set of samples on the implicit surface enables a range of downstream graphics applications including blue noise sampling, neural implicit shape deformation, and many others.

6.1. Sampling Across Representations

Our method is applicable to a variety of surface representations, including not just implicit surfaces but generally anything against which we can cast rays and gather intersections. Figure 12 demonstrates point samples across a collection of these representations. Analytical implicit surfaces can be defined by sweeping and revolution (Figure 12), or authored by an artist (Figure 15). Neural implicit representations (Figure 12) such as Siren [SMB*20], NeuS [WLL*21], or latent interpolations [LWJ*22] are used in reconstruction and generative modeling. We also include a preliminary example on a Gaussian particle surface [KKLD23] (Figure 12); although there is no precise notion of a surface, our algorithm produces reasonable point sets by casting rays as in [MLMP*24], marking an intersection when the transmittance drops below 50% and casting new rays from the intersection points to obtain all intersections along the same direction.

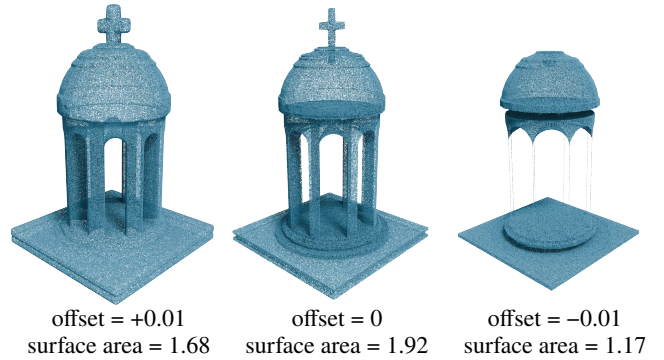


Figure 14: White noise sampling can be straightforward with mesh surfaces, but offset surface sampling is often non-trivial unless converted to an implicit surface. Our method can easily sample offset surfaces, along both positive and negative directions, in addition to the surface defined at the zero level set.

Open Boundaries and Nonmanifold Geometry Open surfaces, variable codimension, and non-manifold structure likewise pose no problem for our approach, so long as we can intersect rays with the surface. Figure 8 shows an example sampling from an open surface

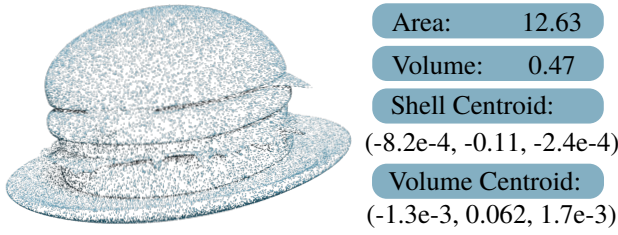


Figure 15: Our method can estimate various shape quantities like surface area, volume, surface centroids, and volumetric centroids, as demonstrated here for this PseudoSDF ShaderToy example of a burger as seen in Figure . ShaderToy PseudoSDF credit to ©Xor (CC BY-NC-SA 3.0)

by tracing an unsigned distance function, while the swept surface in Figure 12 is non-manifold.

Offset Surfaces Even when working with an explicit mesh representation, our approach may be useful to sample from implicit, derived surfaces such as the *offset surface* from a mesh, defined as the shifted surface which is a specific distance away from the input mesh. We can easily cast rays against this offset surface by querying and shifting the distance from the mesh at any point in space; Figure 14 shows one example of sampling points from such an offset surface. In turn this also allows the estimation of derived quantities like surface area of the offset surface, as described below.

6.2. Moment Estimation

Our ray-casting-based sampling can also be used to compute moments of the shape. For the calculations below we assume that the shape is enclosed in the $[-1, 1]^3$ cube. All of these formulas, or variants of them, are well-known, so we will only briefly discuss each one. See Figure 15 for an example of moments estimation on a pseudo-SDF ShaderToy example of a burger.

Surface Area If M random rays in the $[-1, 1]^3$ cube against an enclosed surface S results in K intersections, then a Cauchy-Crofton-like formula for the surface area of S is:

$$A = \int_S 1 d\mathbf{x} = \lim_{M \rightarrow \infty} \frac{12K}{M}. \quad (2)$$

This formula can be derived by taking ratios of Monte Carlo estimates from the Cauchy-Crofton formula between S and $[-1, 1]^3$, to eliminate the proportional constant depending on the ray integration volume [DLI10].

Thin Shell For a surface S , averaging uniform samples directly on S gives a Monte Carlo estimate of its centroid:

$$\mathbf{c}_{\text{shell}} = \frac{1}{A} \int_S \mathbf{x} d\mathbf{x} = \lim_{M \rightarrow \infty} \frac{1}{K} \sum_{i=1}^K \mathbf{x}_i. \quad (3)$$

Volume Treating the shape as enclosing a solid volumetric region Ω , our intersections also give a Cauchy-Crofton-like formula for the total volume. Similar formulas can be derived using the divergence theorem for rays in a single direction [KH10, TCL21].

$$V = \int_{\Omega} 1 d\mathbf{x} = \lim_{M \rightarrow \infty} \frac{6}{M} \sum_{i=1}^M \sigma_i \quad (4)$$

where σ_i is the chord length for the i -th ray (i.e., the total length of the portion(s) of the ray inside the shape). The chord length can be efficiently tracked during consecutive intersection tracing. We can derive Eq. 4 using the fact that the average chord length of rays passing through Ω is $4V/A$ [San76, Maz08], and taking the ratio of volumes between Ω and $[-1, 1]^3$ to eliminate the proportional constant.

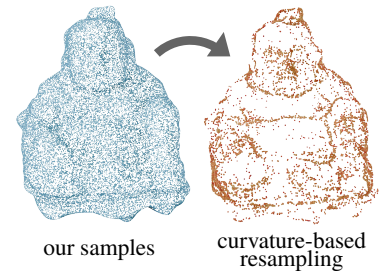
Similarly, we can approximate the volumetric centroid:

$$\mathbf{c}_{\text{solid}} = \frac{1}{V} \int_{\Omega} \mathbf{x} d\mathbf{x} = \frac{\sum_{i=1}^M \sum_{j \in \mathcal{C}_i} \frac{\sigma_{ij}}{2} (\mathbf{a}_{ij} + \mathbf{b}_{ij})}{\sum_{i=1}^M \sum_{j \in \mathcal{C}_i} \sigma_{ij}} \quad (5)$$

where \mathcal{C}_i defines the (possibly empty) set of chords for the i -th ray so that σ_{ij} is the length of the j -th chord of the i -th ray and \mathbf{a}_{ij} and \mathbf{b}_{ij} are that chord's corresponding start and end points. This formula is a straightforward corollary of Eq. 4.

6.3. Curvature-based Resampling

With a uniformly distributed set of samples that covers every part of the surface, one can then estimate per-point metrics such as curvature and loss values for importance sampling or hard sample mining.



Such techniques have shown benefits for tasks like neural implicit surface optimization from images [YWOSH21]. On the right we show points obtained via curvature-based resampling on the intermediate surface during optimization [SMB*20].

6.4. Low-Discrepancy Sequence Sampling

One direct downstream application that requires a uniform set of samples is blue noise sampling via subsampling [Yuk15]. These samples can then be used for texture synthesis, simulation or remeshing. See Figure 2 for an example, where we subsample our uniform samples as blue noise.

One can also replace the uniform ray sampling specified in Section 4.1 with low-discrepancy sequences of rays as proposed in [LYZ*06, LYZ*10]. This does not necessarily yield a corresponding low-discrepancy sequence on the surface, however we find that it indeed improves convergence when estimating moments, as shown in Figure 16.

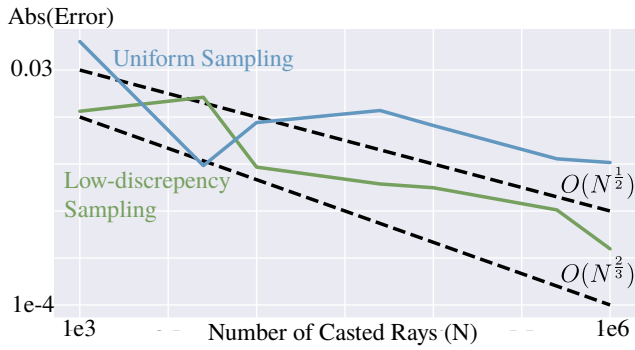
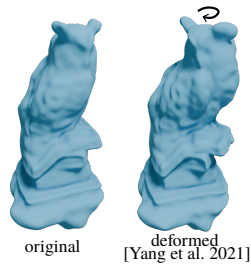


Figure 16: We plot the absolute surface area estimation error with samples acquired from sampled rays using a low-discrepancy sequence as proposed in [LYZ*06, LYZ*10], as well as uniformly sampled rays, as described in Section 4.1. We also plot asymptotes $O(N^{\frac{2}{3}})$ and $O(N^{\frac{1}{2}})$, and observe that using the low-discrepancy sequence results in faster convergence, as pointed out in [LYZ*06].

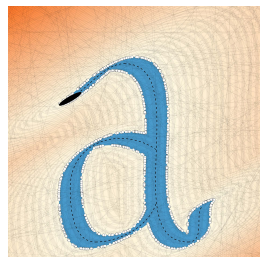
6.5. Neural Implicit Deformation

A set of uniformly distributed samples on a surface is also essential for defining surface loss functions for neural implicit deformation as described in [YBHK21]. The original paper considers a Langevin dynamics-based sampling method which suffers from clumping near high-curvature regions and empirically uses rejection sampling with projection in their method, which we compared to as one of the baselines in Section 5.1. Here we replace the sampling module in the original implementation with our method and show one result on the right.



6.6. Two Dimensional Shapes

While our primary motivation is sampling surfaces in 3D, our algorithm trivially generalizes to 2D. The recent surge of interest in generative vector graphics [YCC*25] and diffusion models for vector fonts [TLA*24] motivates the need for sampling structured shape outlines in the plane. There is already success utilizing implicit functions for these tasks [RZW*21]. The inset figure shows a classic Metafont ‘a’ — defined as the Minkowski sum of a rotated ellipse along a spline [Knu86] — treated as an implicit function, whose boundary is uniformly sampled with our approach restricted to 2D, zoom to see samples. Generalization to 4D and beyond is also exciting to consider, though the average chord length formulas [San76, Maz08] indicate decreasing efficiency with dimension.



7. Conclusions

This work considers the problem of uniformly randomly sampling surfaces without an explicit mesh representation, such as implicit surfaces. We presented a solution via randomly sampled rays cast in space and finding ray intersections with the implicit level set. While our method is grounded in existing theory and prior explorations, we evaluate it on a wide variety of implicit surfaces including modern neural representations. By leveraging the power of modified sphere tracing, we show its advantages over common baselines such as sampling on an extracted mesh, rejection sampling, and a Markov Chain Monte Carlo-based method. We explore many downstream applications that extend from our method or benefit from a uniformly-distributed set of samples on a surface.

Limitations and Future Work For shapes with extremely thin sparse features which occupy only a small fraction of their bounding box, sampling may become inefficient as most rays do not intersect the geometry. All sampling methods struggle with this case in some form or another: grid-based approaches such as marching cubes may be entirely alias and miss these features, while MCMC-type methods (e.g., [Chi22]) may see proposal rejections increase leading to undersampling and over-duplicating. We believe the behavior of our method is often preferential in this case, in that its sampling is still accurate but merely less efficient. There are many avenues to improve the efficiency of our method as well, by using tighter bounding primitives and/or by using bounding primitives that support faster ray sampling, such as spheres.

Many of our examples leverage sphere tracing of SDF-like implicit functions; this fast tracing algorithm is only possible when the function has a known (or estimable) Lipschitz constant, otherwise this method must fall back on dense marching or specialized searches (e.g., [GYBC24]). Other common implicits such as truncated signed distance fields or neural fields [XTS*21] that are constructed with accompanying space-skipping data-structures are readily incorporated into our approach.

Some implicit formulations might model hard surfaces as a 0-1 binary occupancy field or a similar density formulation; such fields are difficult to efficiently trace and sample from, as there is little information in the vicinity of the interface. The stochastic framework for Poisson surface reconstruction [SJ22] provides one promising starting point, modeling the smoothed near-surface uncertainty which is inherently present.

There are still many interesting use cases that we would like to explore as future work. For example, certain simulators require estimation of intersection volumes between two shapes [TLP*18], for which our shape quantity estimation property can be helpful. One could also go one dimension higher, and instead of sampling points via random ray casting, sample curves via random plane intersections as studied in [BM20].

Acknowledgments

We thank Derek Liu and Silvia Sellán for providing codes. Our research is funded in part by NSERC Discovery (RGPIN-2022-04680), the Ontario Early Research Award program, the Canada Research Chairs Program, a Sloan Research Fellowship, the DSI Catalyst Grant program and gifts by Adobe Inc.

References

- [AHY*19] ATZMON M., HAIM N., YARIV L., ISRAELOV O., MARON H., LIPMAN Y.: Controlling neural level sets. *Advances in Neural Information Processing Systems* 32 (2019). 2
- [BM20] BREIDING P., MARIGLIANO O.: Random points on an algebraic manifold. *SIAM Journal on Mathematics of Data Science* 2, 3 (2020), 683–704. 12
- [Bri07] BRIDSON R.: Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches* 10, 1 (2007), 1. 2, 4
- [Chi22] CHIU E.: *Uniform Sampling over Level Sets*. Master's thesis, Massachusetts Institute of Technology, 2022. 3, 8, 9, 12
- [CYAE*20] CAI R., YANG G., AVERBUCH-ELOR H., HAO Z., BELONGIE S., SNAVELY N., HARIHARAN B.: Learning gradient fields for shape generation. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16* (2020), Springer, pp. 364–381. 3
- [DBD*22] DARMON F., BASCLE B., DEVAUX J.-C., MONASSE P., AUBRY M.: Improving neural implicit surfaces geometry with patch warping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 6260–6269. 3
- [DHJM08] DETWILER J. A., HENNING R., JOHNSON R. A., MARINO M. G.: A generic surface sampler for monte carlo simulations. *IEEE Transactions on Nuclear Science* 55, 4 (Aug. 2008), 2329–2333. URL: <http://dx.doi.org/10.1109/TNS.2008.2001063>, doi:10.1109/tns.2008.2001063. 2, 3, 5
- [DLI10] DAUGMAUDIS J. V., LAURYNENAS A., IVANAUSKAS F.: The influence of bounding surface on the precision of the cauchy–crofton method. *Lietuvos matematikos rinkiny* 51, proc. LMS (Dec. 2010), 250–255. URL: <https://www.journals.vu.lt/LMR/article/view/17823>, doi:10.15388/LMR.2010.46.11
- [Fra63] FRANKLIN J. N.: Deterministic simulation of random processes. *Mathematics of Computation* 17, 81 (1963), 28–59. 4
- [GGPP20] GALIN E., GUÉRIN E., PARIS A., PEYTAVIE A.: Segment tracing using local lipschitz bounds. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 545–554. 6
- [GYBC24] GILLESPIE M., YANG D., BOTSCH M., CRANE K.: Ray tracing harmonic functions. *ACM Trans. Graph* 43, 4 (2024). 2, 12
- [Har96] HART J. C.: Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545. 2, 6
- [HCHH24] HUANG S.-S., CHEN G., HENG C. L., HUANG H.: Neural indicator: implicit surface reconstruction from neural indicator priors. In *Forty-first International Conference on Machine Learning* (2024). 2, 3
- [HNSS*24] HÄRENSTAM-NIELSEN L., SANG L., SAROHA A., ARASLANOV N., CREMERS D.: Diffcd: A symmetric differentiable chamfer distance for neural implicit surface fitting. In *European Conference on Computer Vision* (2024), Springer, pp. 432–447. 2, 3, 8, 9
- [KH10] KHOSRAVIFARD A., HEMATIYAN M. R.: A new method for meshless integration in 2d and 3d galerkin meshfree methods. *Engineering Analysis with Boundary Elements* 34, 1 (2010), 30–40. 11
- [KKCF13] KING A., KULLA C., CONTY A., FAJARDO M.: Bssrdf importance sampling. In *ACM SIGGRAPH 2013 Talks*. 2013, pp. 1–1. 2
- [KKLD23] KERBL B., KOPANAS G., LEIMKÜHLER T., DRETTAKIS G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1. 10
- [Knu86] KNUTH D. E.: *The METAFONT book*, vol. C of *Computers and Typesetting*. Addison-Wesley, Reading, Massachusetts, 1986. 12
- [LC98] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 1998, pp. 347–353. 3
- [LLK*22] LI L., LIU Q., KORBA A., YUROCHKIN M., SOLOMON J.: Sampling with mollified interaction energy descent. *arXiv preprint arXiv:2210.13400* (2022). 3
- [LWJ*22] LIU H.-T. D., WILLIAMS F., JACOBSON A., FIDLER S., LITANY O.: Learning smooth neural functions via lipschitz regularization. In *ACM SIGGRAPH 2022 Conference Proceedings* (2022), pp. 1–13. 4, 9, 10
- [LWMB03] LI X., WANG W., MARTIN R. R., BOWYER A.: Using low-discrepancy sequences and the crofton formula to compute surface areas of geometric models. *Computer-Aided Design* 35, 9 (2003), 771–782. 2, 3
- [LYZ*06] LIU Y.-S., YONG J.-H., ZHANG H., YAN D.-M., SUN J.-G.: A quasi-monte carlo method for computing areas of point-sampled surfaces. *Computer-Aided Design* 38, 1 (2006), 55–68. 3, 11, 12
- [LYZ*10] LIU Y.-S., YI J., ZHANG H., ZHENG G.-Q., PAUL J.-C.: Surface area estimation of digitized 3d objects using quasi-monte carlo methods. *Pattern Recognition* 43, 11 (2010), 3900–3909. 3, 11, 12
- [Maz08] MAZZOLO A.: On the generalization of the average chord length. *Annals of Nuclear Energy* 35, 3 (2008), 503–506. URL: <https://www.sciencedirect.com/science/article/pii/S0306454907001843>, doi:https://doi.org/10.1016/j.anucene.2007.06.026. 11, 12
- [MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15. 6
- [MLMP*24] MOENNE-LOCOCOZ N., MIRZAEI A., PEREL O., DE LUTIO R., MARTINEZ ESTURO J., STATE G., FIDLER S., SHARP N., GOJCIC Z.: 3d gaussian ray tracing: Fast tracing of particle scenes. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–19. 2, 10
- [MPZ14] MYLES A., PIETRONI N., ZORIN D.: Robust field-aligned global parametrization. *ACM Trans. Graph.* 33, 4 (2014), 135–1. 7, 8, 9, 15
- [ÖAG10] ÖZTIRELI A. C., ALEXA M., GROSS M.: Spectral sampling of manifolds. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 1–8. 4
- [PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 165–174. 2
- [PPK16] PALAIS R., PALAIS B., KARCHER H.: Pointclouds: Distributing points uniformly on a surface. *arXiv preprint arXiv:1611.04690* (2016). 2, 4, 14
- [Qui09] QUINN J. A.: *Low-discrepancy point sampling of 2D manifolds for visual computing*. Cardiff University (United Kingdom), 2009. 3
- [Rub87] RUBIN D. B.: The calculation of posterior distributions by data augmentation: Comment: A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: The sir algorithm. *Journal of the American Statistical Association* 82, 398 (1987), 543–546. URL: <http://www.jstor.org/stable/2289460>. 5
- [RWCS05] ROVIRA J., WONKA P., CASTRO F., SBERT M.: Point sampling with uniformly distributed lines. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005*. (2005), IEEE, pp. 109–118. 3
- [RZW*21] REDDY P., ZHANG Z., WANG Z., FISHER M., JIN H., MITRA N. J.: A multi-implicit neural representation for fonts. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6–14, 2021, virtual* (2021), Ranzato M., Beygelzimer A., Dauphin Y. N., Liang P., Vaughan J. W., (Eds.), pp. 12637–12647. URL: <https://proceedings.neurips.cc/paper/2021/hash/6948bd44c91acd2b54ecdd1b132f10fb-Abstract.html>. 12

- [San76] SANTALÓ L. A.: *Integral geometry and geometric probability*. 1976. 4, 9, 11, 12
- [SD11] SCHLÖMER T., DEUSSEN O.: Accurate spectral analysis of two-dimensional point sets. *J. Graphics, GPU, & Game Tools* 15, 3 (2011), 152–160. URL: <https://doi.org/10.1080/2151237X.2011.609773>, doi:10.1080/2151237X.2011.609773. 2
- [SDF97] STOL J., DE FIGUEIREDO L. H.: Self-validated numerical methods and applications. In *Monograph for 21st Brazilian Mathematics Colloquium, IMPA, Rio de Janeiro. Citeseer* (1997), vol. 5, Citeseer, p. 122. 2
- [SJ00] SCHAUFLEER G., JENSEN H. W.: Ray tracing point sampled geometry. In *Rendering Techniques 2000: Proceedings of the Eurographics Workshop in Brno, Czech Republic, June 26–28, 2000 11* (2000), Springer, pp. 319–328. 3
- [SJ22] SELLÁN S., JACOBSON A.: Stochastic poisson surface reconstruction. *ACM Trans. Graph.* 41, 6 (2022), 227:1–227:12. URL: <https://doi.org/10.1145/3550454.3555441>, doi:10.1145/3550454.3555441. 12
- [SMB*20] SITZMANN V., MARTEL J., BERGMAN A., LINDELL D., WETZSTEIN G.: Implicit neural representations with periodic activation functions. *Advances in neural information processing systems* 33 (2020), 7462–7473. 9, 10, 11
- [Sul12] SULLIVAN J. M.: Pleasing shapes for topological objects. In *Mathematics and Modern Art: Proceedings of the First ESMA Conference, held in Paris, July 19–22, 2010* (2012), Springer, pp. 153–165. 9
- [Tal05] TALBOT J. F.: *Importance resampling for global illumination*. Brigham Young University, 2005. 5, 14
- [TCL21] TRUSTY T., CHEN H., LEVIN D. I.: The shape matching element method: Direct animation of curved surface models. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–14. 11
- [TLA*24] THAMIZHARASAN V., LIU D., AGARWAL S., FISHER M., GHARBI M., WANG O., JACOBSON A., KALOGERAKIS E.: Vecfusion: Vector font generation with diffusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16–22, 2024*, IEEE, pp. 7943–7952. URL: <https://doi.org/10.1109/CVPR52733.2024.00759>, doi:10.1109/CVPR52733.2024.00759. 12
- [TLP*18] TONG J., LI Q., PENG W., FANG Y., WU Y.: Computation of intersection volume using discrete quadrature algorithm. *Optik* 158 (2018), 725–736. 12
- [WLL*21] WANG P., LIU L., LIU Y., THEOBALT C., KOMURA T., WANG W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689* (2021). 2, 3, 9, 10
- [XTS*21] XIE Y., TAKIKAWA T., SAITO S., LITANY O., YAN S., KHAN N., TOMBARI F., TOMPKIN J., SITZMANN V., SRIDHAR S.: Neural fields in visual computing and beyond. *CoRR abs/2111.11426* (2021). URL: <https://arxiv.org/abs/2111.11426>, arXiv:2111.11426. 12
- [YBHK21] YANG G., BELONGIE S., HARIHARAN B., KOLTUN V.: Geometry processing with neural fields. *Advances in Neural Information Processing Systems* 34 (2021), 22483–22497. 2, 3, 8, 9, 12
- [YCC*25] YANG Y., CHENG W., CHEN S., ZENG X., ZHANG J., WANG L., YU G., MA X., JIANG Y.-G.: Omnisvg: A unified scalable vector graphics generation model, 2025. URL: <https://arxiv.org/abs/2504.06263>, arXiv:2504.06263. 12
- [YDL*14] YEH H.-Y. C., DENNY J., LINDSEY A., THOMAS S., AMATO N. M.: Umarm: Uniformly sampling the medial axis. In *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), IEEE, pp. 5798–5803. 3
- [YGKL21] YARIV L., GU J., KASTEN Y., LIPMAN Y.: Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems* 34 (2021), 4805–4815. 2
- [Yuk15] YUKSEL C.: Sample elimination for generating poisson disk sample sets. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 25–32. 2, 4, 11
- [YWOSH21] YIFAN W., WU S., OZTIRELI C., SORKINE-HORNUNG O.: Iso-points: Optimizing neural implicit surfaces with hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 374–383. 2, 3, 11
- [YWW14] YAN D.-M., WALLNER J., WONKA P.: Unbiased sampling and meshing of isosurfaces. *IEEE transactions on visualization and computer graphics* 20, 11 (2014), 1579–1589. 3

Appendix A: Proof of Resampling Formulation

Here we show that the ray resampling method described in Section 3.3 produces uniformly distributed samples over a surface S . Before stating the theorem, we define \mathbf{L} as a compact set of rays, which contains all rays in \mathbb{R}^3 that intersect S . Now, we wish to prove the following:

Theorem 1 Given a set of uniformly M distributed rays $L = \{\ell_i\}$ sampled from \mathbf{L} , where each ℓ_i intersects S k_i times, resampling N rays from L with probability proportional to k_i and uniformly selecting one intersection per resampled ray produces a set of points $X = \{x_j\}$ that is uniformly distributed over S .

Proof We will prove the theorem by showing that X is 1-equidistributed; that is, we can build a consistent Monte Carlo estimator of integrals over S using X and uniform contribution weights. We resample rays by following Resampled Importance Sampling (RIS) [Tal05], using a target function of $q(\ell_i) = k_i$ and noting that the base distribution is $p = p(\ell_i) = 1/|\mathbf{L}|$. We assign a resampling weight $w_i = \frac{q_i}{p}$ to each ℓ_i , and resample N rays proportional to w_i . From these resampled rays $L_r = \{\ell_j\}$, we uniformly select an intersection with S , \mathbf{x}_j , with probability $1/k_j$. Through an extension of the Cauchy-Crofton formula for evaluating integrals over S [PPK16], we know that in \mathbb{R}^3 , $\int_S f(x) dx = \frac{1}{2\pi} \int_{\mathbf{L}} \sum_{i=1}^{k(\ell)} f(x_i) d\ell$, i.e., an integral over surfaces can be converted to an integral over rays by summing the integrand over each of the $k(\ell)$ intersections ℓ makes with S , denoted x_i . Using RIS, the Monte Carlo estimator for an arbitrary function f over S is then

$$\begin{aligned} I &= \frac{1}{2\pi} \frac{1}{N} \sum_j \frac{1}{1/k_j} \frac{f(x_j)}{q(\ell_j)} \left(\frac{1}{M} \sum_i w_i \right) \\ &= \frac{1}{2\pi} \frac{1}{N} \sum_j f(x_j) |\mathbf{L}| \frac{\sum_i k_i}{M} \end{aligned}$$

From [PPK16], we know that as $M \rightarrow \infty$, $\frac{\sum_i k_i}{M} \rightarrow \frac{2\pi|S|}{|\mathbf{L}|}^\dagger$, so the equation simplifies to $I = \frac{1}{N} \sum_j |S| f(x_j)$ as $M \rightarrow \infty$, which is the desired result. \square

Appendix B: Analysis: Evaluation Count vs. Surface Area

Besides analyzing the average number of samples per casted ray, we also empirically analyzed the relationship between the total

[†] The proof of this result in [PPK16] has an error where they omit a factor of $\frac{2}{|\mathbf{L}|}$, though a corrected proof follows essentially the same argument presented in the paper, by applying the Cauchy-Crofton formula and accounting for $|\mathbf{L}|$ in Monte Carlo estimates over \mathbf{L} .

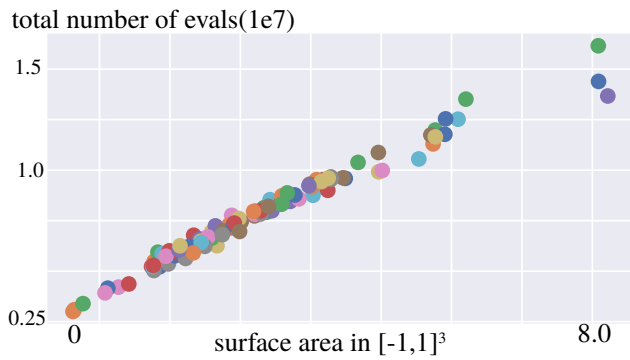


Figure 17: We ran our method to sample points with 50,000 rays on 114 shapes from a dataset of meshes [MPZ14], and plotted the total number of evaluations required and the ground truth surface area.

function evaluation cost of our method and the surface area of the shape being sampled. Not surprisingly, as shown in Figure 17, it also has an approximately linear relationship as sphere tracing-like methods are most costly near the surface.