Communications on Applied Mathematics and Computation
https://doi.org/10.1007/s42967-023-00318-1

ORIGINAL PAPER

Nearest Neighbor Sampling of Point Sets Using Rays

Liangchen Liu[1] · Louis Ly[2] · Colin B. Macdonald[3] · Richard Tsai[1,2]

Received: 8 November 2022 / Revised: 13 September 2023 / Accepted: 13 September 2023

Abstract
We propose a new framework for the sampling, compression, and analysis of distributions of point sets and other geometric objects embedded in Euclidean spaces. Our approach involves constructing a tensor called the RaySense sketch, which captures nearest neighbors from the underlying geometry of points along a set of rays. We explore various operations that can be performed on the RaySense sketch, leading to different properties and potential applications. Statistical information about the data set can be extracted from the sketch, independent of the ray set. Line integrals on point sets can be efficiently computed using the sketch. We also present several examples illustrating applications of the proposed strategy in practical scenarios.

Keywords Point clouds · Sampling · Classification · Registration · Deep learning · Voronoi cell analysis

Mathematics Subject Classification 68T09 · 65D19 · 68T07 · 65D40

Liangchen Liu, Colin B. Macdonald, and Richard Tsai contributed equally to this work.

✉ Liangchen Liu
lcliu@utexas.edu

Louis Ly
louisly@utexas.edu

Colin B. Macdonald
cbm@math.ubc.ca

Richard Tsai
ytsai@math.utexas.edu

[1] Department of Mathematics, The University of Texas at Austin, 2515 Speedway, Austin, TX 78712, USA

[2] Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, 201 E 24th St, Austin, TX 78712, USA

[3] Department of Mathematics, University of British Columbia, 1984 Mathematics Rd, Vancouver, BC V6T 1Z2, Canada

1 Introduction

The comparison and analysis of objects in d-dimensional Euclidean spaces are fundamental problems in many areas of science and engineering, such as computer graphics, image processing, machine learning, and computational biology, to name a few.

When comparing objects in Euclidean spaces, one usually assumes that the underlying objects are solid or continuous. Typical examples include data manifolds and physical or probabilistic density representations.

One commonly used approach is the distance-based comparison, which involves measuring the distance between two objects using metrics such as the Euclidean distance, Manhattan distance, or Mahalanobis distance [35]. When the underlying object can be viewed as distributions, optimal transport [42, 59], utilizing the Wasserstein distance, is also a popular choice.

However, in general, distance-based comparisons overlook helpful geometric information about the underlying objects, which turns out to be useful in many applications. This is because intrinsic geometric features such as curvature and volume are invariant to rotations and translations. Shape-based information is also suitable for comparing objects of different sizes or resolutions. Therefore, comparison techniques using geometric features are favorable choices in many scenarios, such as the object recognition, classification, and segmentation.

One simple example illustrating such an idea is the Monte-Carlo rejection sampling technique used to approximate the volume of an object. This technique considers the collision of 0-dimensional objects (points) with the target object. This is a specific example of a collision detection algorithm [32], which is popular in computer graphics and computational geometry communities [21]. The idea of collisions is also considered in the field of integral geometry [49], where one investigates the collision probability of certain affine subspaces with the target data manifold to deduce information about the manifold.

Furthermore, in integral geometry, one considers integral transforms on the underlying objects. Typical examples are the X-ray transform [55] and the Radon transform [18, 46]. These transforms can provide a more compact and informative representation than the original data. For example, one can recover spectral information from the X-ray transform or reconstruct the original object through an inverse Radon transform.

However, the aforementioned techniques generally rely on the assumption that the underlying object is solid or continuous. With the prevalence of big data and advancements in sensing technology, such as the LiDAR, the analysis and comparison of the point cloud data (which consists of a set of points in some d-dimensional Euclidean spaces) have gained increasing attention, yet they pose challenges to classical approaches.

We propose a novel method for sampling, comparing, and analyzing point clouds, or other geometric objects, embedded in high-dimensional space. We call our approach "RaySense" because it "senses" the structure of the object Γ by sending line segments through the ambient space occupied by Γ and recording some functions of the nearest neighbors in Γ to points on the line segment. Motivated by the X-ray transform, we will refer to the oriented line segments as "*rays*". We can then work with this sampled data as a "sketch" of the original object Γ, which can be a point cloud, triangulated surface, volumetric representation, etc. A visualization of the proposed method applied to 3-dimensional (3D) point clouds is provided in Fig. 1.

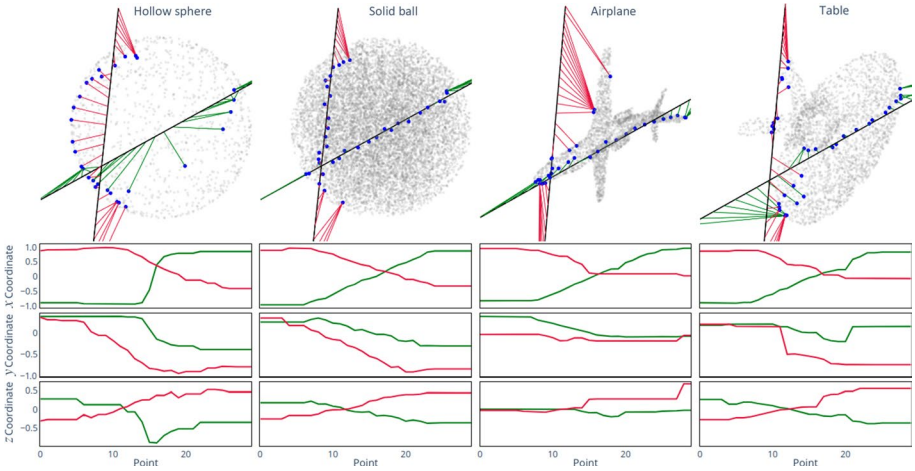


Fig. 1 RaySense sketches using 30 sample points per ray. Row 1: visualization of two rays (black) through points sampled from various objects (gray). Closest point pairs are shown in green and red. Rows 2–4: the x, y, and z coordinates of the closest points to the ray

Our method incorporates several ideas mentioned above as follow.

- In the context of integral geometry, we also consider using affine subspaces to detect the underlying geometry. To overcome the discontinuity from the object representation relative to the topology of the ambient space (as is the case with point clouds), we search for nearest neighbors within the representation. To prevent the computational cost of high-dimensional operations in practice, we work with a low-dimensional affine subspace. In this paper, we use 1-dimensional (1D) lines.
- In the context of inverse problems, our observed data consist of some points on Γ obtained by the closest point projection of points on the rays. This is somewhat analogous to seismic imaging, where designated points on each ray correspond to geophones that record the first arrival time of waves from known sources.
- By using a fixed number of rays and sampling a fixed number of points along a ray, the resulting tensor from our method, the RaySense sketch, is of fixed size, even for point clouds with different cardinalities. Then, metrics suitable for comparing tensors are available for use, or one could consider other distance-based approaches to compare different RaySense sketches.

In this work, we will focus exclusively on point clouds since it is one of the most challenging data representations for many algorithms. When the object is a point cloud, finding the RaySense samples is straightforward via discrete nearest-neighbor searches. There are computationally efficient algorithms for performing nearest-in-Euclidean-distance neighbor searches. Examples include tree-based algorithms [2], grid-based algorithms [58], and building an indexing structure [22]. In very high dimensions, one may also consider randomized nearest neighbor search algorithms such as [19, 23], or certain dimension reduction approaches.

The remaining paper is organized as follows: in Sect. 2, we provide a detailed description of the RaySense method; in Sect. 3, we present various properties of the RaySense

method, along with theoretical analysis, including line integral approximations and salient point sampling; finally, in Sect. 4, we demonstrate that the concept of RaySense can be applied to many different downstream tasks, including the salient point/outlier detection, point cloud registration, and point cloud classification.

1.1 Related Work

In this section, we provide a more detailed discussion of the existing literature to contextualize our approach.

Integral geometry

In the field of Integral Geometry [49], one uses the probability of the intersection of affine subspaces of different dimensions with the target data manifold to deduce information about the manifold itself. For example, in the classic problem of Buffon's needle, one determines the length of a dropped needle by investigating the probability of intersections with strips on a floor. Similarly, Crofton's formula connects the length of a 2-dimensional (2D) plane curve with the expected number of intersections with random lines. In these cases, the interaction information obtained from the "sensing" affine subspaces is binary: yes or no. One thus has a counting problem: how frequently affine subspaces intersect with the data manifold. From these probabilities, geometric information about the manifold can be extracted, relying on a duality between the dimensions of the "sensing" subspaces and the Hausdorff dimensions of the data set; see e.g., [17, 26]. Nevertheless, such approaches may be inefficient in practical computations.

Our idea of using rays is perhaps most-closely related to the X-ray transform, which coincides with the Radon Transform in two dimensions [39]. In an X-ray transform, one integrates a given real-valued function defined in $\mathbb{R}^d$ along lines, while in the Radon transform, one integrates the given function restricted on hyperplanes of $\mathbb{R}^d$.

We advocate using lines (rays) to record information about the underlying data along each ray, instead of accumulating a scalar or binary "yes/no" information over each rays. In this paper, we collect points in the data set closest to the points on the rays, along with the values of some function at those points. With such data, we can compute approximate line integrals and relate our method to the X-ray transform.

Computer vision

From the perspective of the computer vision community, our method can be considered as a shape descriptor, mapping from 3D point sets to a more informative feature space where point sets can be easily compared. Generally, descriptors aim to capture statistics related to local and global features. See [24] for a survey. More recently, researchers have combined shape descriptors with machine learning [15, 47, 54, 64]. But all these works focus primarily for point sets in 3D, RaySense applies more generally to data in arbitrary dimensions.

Some methods use machine learning directly on the point set to learn features for specific tasks, such as classification [1, 27, 31, 44, 45, 53, 60, 61, 65]. PointNet [44] pioneered deep learning on point clouds by applying independent operations on each point and aggregating features using a symmetric function. Building upon that, other architectures [45, 52] exploit neighboring information to extract local descriptors. SO-Net [30] uses self-organizing maps to hierarchically group nodes while applying fully-connected layers for feature extraction. PCNN [1] defines an extension and pulling operator similar to the closest point method [33, 48] to facilitate the implementation of regular convolution. DGCNN [61] and PointCNN [31] generalize the convolution operator to point sets.

In contrast, our approach uses the RaySense sketch as input rather than applying machine learning directly to the point set. In Sect. 4.5, we present a deep learning model for 3D point cloud classification based on this idea. Our experiments suggest that the model is very efficient for classification, and the resulting classifiers can be robust against outliers by storing multiple nearest neighbors to points on the ray.

Unsupervised learning methods

Our method, by recording the closest points from the underlying point cloud, can be thought of as a sampling scheme. However, the RaySense sampling is biased towards the "salient" points in the underlying geometry, as will be discussed in Sect. 3.2.1. By further retaining only the most frequently sampled points, RaySense resembles key-point detectors [28] or compression algorithms. The idea of understanding the overall structure of the underlying data through key points is closely related to archetypal analysis [8], which involves finding representative points (archetypes) in a dataset and using a convex combination of these archetypes to represent other points. See also [40] for a recent work on the consistency of archetypal analysis.

Incidentally, [16] also employs the concept of rays in conjunction with spherical volume to approximate the convex hull. Our method can also capture vertices of the convex hull when the rays are sufficiently long, as it will effectively sample points on the portions of the boundary that have relatively large positive curvature.

Farthest Point Sampling (FPS) is a widely-used sampling method in computational geometry and machine learning for selecting a subset of points from a larger dataset with the goal of maximizing their spread; see e.g., [14]. The process begins by randomly picking a point from the dataset, followed by iterative selection of the point farthest from those already chosen, until a desired number of points are selected. This technique is useful for reducing the size of large datasets while preserving their overall structure. However, it can be computationally expensive and may not always yield the optimal solution. In two and three dimensions, assuming that the data distribution is supported in a bounded convex set with a smooth boundary, FPS tends to oversample areas with high curvature.

2 Methods

The essential elements of the proposed sampling strategy include (i) the data set $\Gamma \subset \mathbb{R}^d$; (ii) the nearest neighbor (closest point) projection, P_Γ, to Γ; (iii) a distribution of lines in $\mathbb{R}^d$.

The data set is given by a discrete set of N points, each in $\mathbb{R}^d$:

$$\Gamma \subset \mathbb{R}^d, \qquad \Gamma = \{\boldsymbol{X}_i\}_{i=1}^N, \qquad \boldsymbol{X}_i \in \mathbb{R}^d \tag{1}$$

(later for certain results we will place more assumptions, for example that Γ might be sampled from a density).

Let $\mathcal{L}$ denote a distribution of lines, parameterized by $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ and $\boldsymbol{b} \in \mathbb{R}^d$, let $\boldsymbol{r}(s)$ denote a line in $\mathcal{L}$ parameterized by its length

$$\boldsymbol{r}(s) = \boldsymbol{b} + s\boldsymbol{\theta}.$$

The parameterization gives an orientation to the line, and thereby we refer to $\boldsymbol{r}$ as a ray. Along $\boldsymbol{r}(s)$, we sample from the data set Γ using the nearest neighbor projection

$$\mathcal{P}_\Gamma \boldsymbol{r}(s) \in \underset{\boldsymbol{y}\in\Gamma}{\arg\,min} \|\boldsymbol{r}(s) - \boldsymbol{y}\|_2. \tag{2}$$

In cases of non-uniqueness, we choose arbitrarily.

Using the nearest neighbors, we define various RaySense sampling operators denoted with $\mathcal{S}$ which sample from the point cloud Γ into some "feature space" $\mathbb{X} \subset \mathbb{R}^c$. The simplest choice is the feature space of closest points defined next.

Definition 1 The closest point feature space is sampled by

$$\mathcal{S}[\Gamma]: \mathbb{R}^d \to \mathbb{R}^d, \qquad \mathcal{S}[\Gamma](\boldsymbol{r}(s)) = \mathcal{P}_\Gamma \boldsymbol{r}(s), \qquad \boldsymbol{r} \sim \mathcal{L}. \tag{3}$$

One might also be interested in the value of a scalar or vector function (e.g., color or temperature data at each point in the point cloud).

Definition 2 The RaySense sampling operator of a function $g: \Gamma \to \mathbb{R}^c$ is

$$\mathcal{S}[\Gamma, g]: \mathbb{R}^d \to \mathbb{R}^c, \qquad \mathcal{S}[\Gamma, g](\boldsymbol{r}(s)) = g\big(\mathcal{P}_\Gamma \boldsymbol{r}(s)\big), \qquad \boldsymbol{r} \sim \mathcal{L}. \tag{4}$$

Note for the identity function we have $\mathcal{S}[\Gamma, \mathrm{id}] = \mathcal{S}[\Gamma]$.

We will further present examples involving the use of *multiple* nearest neighbors, where the ηth nearest neighbor is

$$\mathcal{P}^\eta_\Gamma(\boldsymbol{r}(s)) := \underset{\substack{\boldsymbol{y}\in\Gamma \\ \mathcal{P}^1_\Gamma \boldsymbol{r}(s), \cdots, \mathcal{P}^{\eta-1}_\Gamma \boldsymbol{r}(s)}}{\arg\,min} \|\boldsymbol{r}(s) - \boldsymbol{y}\|_2 \tag{5}$$

with $\mathcal{P}^1_\Gamma := \mathcal{P}_\Gamma$. We can then capture the ηth nearest neighbor in our sampling.

Definition 3 The RaySense sampling operator of the ηth nearest neighbor is denoted by

$$\mathcal{S}[\Gamma, \eta]: \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}^d, \qquad \mathcal{S}[\Gamma, \eta](\boldsymbol{r}(s)) = \mathcal{P}^\eta_\Gamma \boldsymbol{r}(s), \qquad \boldsymbol{r} \sim \mathcal{L}. \tag{6}$$

The distance and direction from $\boldsymbol{r}(s)$ to $\mathcal{P}_\Gamma \boldsymbol{r}(s)$ is sometimes useful: we sample that information using the vector from $\boldsymbol{r}(s)$ to $\mathcal{P}_\Gamma \boldsymbol{r}(s)$.

Definition 4 The RaySense sampling operator of the vector between a point on the ray and the nearest neighbor in Γ is

$$\mathcal{S}[\Gamma, \hat{1}]: \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}^d, \qquad \mathcal{S}[\Gamma, \hat{1}](\boldsymbol{r}(s)) = \mathcal{P}_\Gamma \boldsymbol{r}(s) - \boldsymbol{r}(s), \qquad \boldsymbol{r} \sim \mathcal{L}, \tag{7a}$$

and more generally

$$\mathcal{S}[\Gamma, \hat{\eta}]: \mathbb{R}^d \times \mathbb{N} \to \mathbb{R}^d, \qquad \mathcal{S}[\Gamma, \hat{\eta}](\boldsymbol{r}(s)) = \mathcal{P}^\eta_\Gamma \boldsymbol{r}(s) - \boldsymbol{r}(s), \qquad \boldsymbol{r} \sim \mathcal{L}. \tag{7b}$$

We can augment the feature space by combining these various operators, concatenating the output into a vector $\mathcal{X} \in \mathbb{R}^c$. We indicate this "stacking" with a list notation in $\mathcal{S}$, for example the first three closest points could be denoted

$$\mathcal{S}\big[\Gamma, [1,2,3]\big](\boldsymbol{r}(s)) = \begin{bmatrix} \mathcal{P}^1_\Gamma \boldsymbol{r}(s) \\ \mathcal{P}^2_\Gamma \boldsymbol{r}(s) \\ \mathcal{P}^3_\Gamma \boldsymbol{r}(s) \end{bmatrix}, \quad \boldsymbol{r} \sim \mathcal{L}, \tag{8}$$

or the closest point, its vector from the ray, and the value of a function g could all be denoted

$$\mathcal{S}\big[\Gamma, [1,\hat{1},g]\big](\boldsymbol{r}(s)) = \begin{bmatrix} \mathcal{P}_\Gamma \boldsymbol{r}(s) \\ \mathcal{P}_\Gamma \boldsymbol{r}(s) - \boldsymbol{r}(s) \\ g\big(\mathcal{P}_\Gamma \boldsymbol{r}(s)\big) \end{bmatrix}, \quad \boldsymbol{r} \sim \mathcal{L}. \tag{9}$$

These stacked feature spaces are used in the line integral approximation Sect. 3.1.3 and in our neural network Sect. 4.5.

In summary, a RaySense sketch $\mathcal{S}[\Gamma, \cdots]$ depends on nearest neighbors in the data set Γ, and operates on a ray from the distribution $\mathcal{L}$. It maps a ray $\boldsymbol{r}(\cdot)$ to a piecewise curve in the chosen feature space $\mathbb{X}$.

2.1 Discretization

We propose to work with a discretized version of the operator, which we shall call a RaySense sketch tensor. First, we take m i.i.d. samples from the distribution $\mathcal{L}$ and define m rays correspondingly. We consider n_r uniformly-spaced points along each ray, with corresponding spacing δr. We then work with a finite segment of each line (for example, $0 \leqslant s \leqslant 1$). Appendix A shows some different distributions of lines, and details for choosing segments from them. With $\boldsymbol{r}_{i,j}$ denoting the jth point on the ith ray, we define the *discrete RaySense sketch* of Γ in the closest point feature space as

$$\mathcal{S}_{m,\delta r}[\Gamma; \mathcal{L}]_{i,j} := \mathcal{P}_\Gamma \boldsymbol{r}_{i,j}, \tag{10a}$$

or the discrete RaySense sketch of a function g:

$$\mathcal{S}_{m,\delta r}[\Gamma, g; \mathcal{L}]_{i,j} := g\big(\mathcal{P}_\Gamma \boldsymbol{r}_{i,j}\big) \tag{10b}$$

(and similarly for the various more-general feature spaces mentioned earlier).

Thus, $\mathcal{S}_{m,\delta r}$ is an array with m entries, where each entry is an array of n_r vectors in $\mathbb{R}^c$; an $m \times n_r \times c$ tensor. We will also denote these tensors as "$\mathcal{S}(\Gamma)$" and "$\mathcal{S}(\Gamma, g)$" when m and δr are not the focus of the discussion. In any case, we regard $\mathcal{S}(\Gamma)$ as a "sketch" of the point cloud Γ in a chosen feature space $\mathbb{X}$.

2.2 Operations on RaySense Sketches

In this paper, we will analyze the outcomes of the following operations performed on the RaySense sketches. After discretization, each operation can be expressed as one or more summations performed on the sketches.

2.2.1 Histograms

One simple operation is to aggregate the sampled values for the feature space and count their corresponding sampling frequencies, which results in a histogram demonstrating the

discrete distribution of the corresponding features. For example, when considering the closest point feature space $\mathcal{S}[\Gamma]$, the value of a bin in the histogram, denoted by H_k, represents the number of times $\boldsymbol{x}_k \in \Gamma$ being sampled by RaySense:

$$H_k := H\big(\boldsymbol{x}_k; \mathcal{S}_{m,\delta r}[\Gamma]\big) = \sum_{i=1}^{m} \sum_{j=1}^{n_r} \chi_{\boldsymbol{x}_k}\big(\mathcal{S}[\Gamma]_{i,j}\big), \tag{11}$$

where $\chi_{\boldsymbol{x}_k}$ denotes the indicator function for the coordinates of $\boldsymbol{x}_k$. The histograms discard locality information in the sketch, treating it essentially as a weighted subsampling of Γ, as the aggregation process involves combining the values without considering the specific rays from which they originated. Section 3.2.2 further discusses the properties of the histogram.

2.2.2 Line Integrals

In many applications, it is useful to maintain some "locality". One such operation is the line integral along each ray $\int_0^1 g(\boldsymbol{r}(s))\mathrm{d}s$. However, for point cloud data, exact information of g along the ray $\boldsymbol{r}(s)$ is not accessible, we instead consider the integral along the associated path in feature space, $\int_0^1 \mathcal{S}[\Gamma, g](\boldsymbol{r}(s))\mathrm{d}s$, which we call *a RaySense integral*. We investigate the relationship between the two in Sect. 3.1.3.

In the discrete setting, we can use a simple Riemann sum quadrature scheme to approximate the RaySense integral along the ith ray:

$$\int_0^1 \mathcal{S}[\Gamma, g]\big(\boldsymbol{r}_i(s)\big)\mathrm{d}s \approx \sum_{j=1}^{n_r} \mathcal{S}[\Gamma, g]_{i,j}\, \delta r. \tag{12a}$$

Or in most of our examples, we approximate using the Trapezoidal Rule:

$$\int_0^1 \mathcal{S}[\Gamma, g]\big(\boldsymbol{r}_i(s)\big)\mathrm{d}s \approx \sum_{j=1}^{n_r} w_j\, \mathcal{S}[\Gamma, g]_{i,j}\, \delta r \quad \text{with weights } w = \left\langle \tfrac{1}{2}, 1, \cdots, 1, \tfrac{1}{2} \right\rangle, \tag{12b}$$

and thus quadrature errors, typically $\mathcal{O}(\delta r^2)$, are incurred [57]. Unpacking the notation, we can rewrite this as

$$\sum_{j=1}^{n_r} w_j g(\mathcal{P}_\Gamma \boldsymbol{r}_i(s_j))\delta r. \tag{12c}$$

2.2.3 Convolutions

Similar to line integrals, we will compute convolutions along the rays

$$\big(K * \mathcal{S}[\Gamma, \cdots](\boldsymbol{r})\big)(t) = \int_{-\infty}^{\infty} K(s)\mathcal{S}[\Gamma, \cdots](\boldsymbol{r}(t-s))\mathrm{d}s,$$

where K is some compactly-supported kernel function and $\mathcal{S}[\Gamma, \cdots]$ is one of the RaySense sampling operators. In the discrete setting, this can be written as weighted sums of

$\mathcal{S}[\Gamma, \cdots]$, or as band-limited matrix multiplication. In Sect. 4.5, the discrete weights associated with discrete convolutions are the parameters of a neural network model.

Note unlike the previous cases, for non-symmetric kernels the orientation of the rays matters.

2.3 Comparing Data Sets

Since the sketch for any point set is a fixed-size tensor storing useful feature information, one might compare two point sets (of potentially different cardinalities) via comparing the RaySense sketches.

A natural idea is to choose a suitable metric to define the distances between the RaySense sketch tensors.

The Frobenius norm of the sketch tensor is suitable if the sketches contain the distance and the closest point coordinates. For data sampled from smooth objects, such information along each ray is piecewise continuous. Thus, if the sketches are generated using the same set of sampling rays, one may compare the RaySense sketches of different data sets using the Frobenius norm.

Wasserstein distances are more appropriate for comparison of histograms of the RaySense data, especially when the sketches are generated by different sets of random rays. The normalized histograms can be regarded as probability distributions. In particular, notice (Fig. 2) that RaySense histograms tend to have "spikes" that correspond to the salient points in the data set; ℓ^2 distances are not adequate for comparing distributions with such features.

Here we briefly describe the Wasserstein-1 distance, or Earth mover's distance, that we used in this paper. Let (X, μ_1) and $(\tilde{X}, \mu_2)$ be two probability spaces and F and G be the cumulative distribution functions of μ_1 and μ_2, respectively. The Wasserstein-1 distance is defined as

$$W_1(\mu_1, \mu_2) := \int_{\mathbb{R}} |F(t) - G(t)| \mathrm{d}t.$$

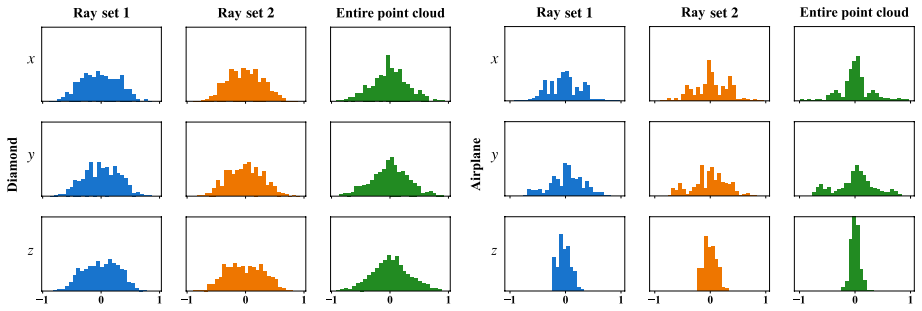


Fig. 2 Histogram of coordinates from two point sets. Columns 1 and 2 correspond to two different sets of rays, each containing 256 rays and 64 samples per ray. These histograms are similar for the same object and different for different objects. Column 3 corresponds to the entire point cloud; these differ from the RaySense histograms especially for the airplane which is not as regular as the diamond

Neural networks One can consider using a properly designed and trained *neural network*. In Sect. 4.5, we present a neural network model, RayNN, for comparing point clouds in three dimensions based on RaySense sketches.

3 Properties of RaySense

In this section, we introduce several notable properties associated with our proposed method. Specifically, we begin by analyzing the characteristics of the method when utilizing a single ray, i.e., $m = 1$. We subsequently proceeded to discuss how the RaySense sketch, resulting from using multiple rays, effectively utilizes and inherits the identified traits. Through these analyses, we aim to provide a more rigorous and comprehensive understanding of the properties and potential applications of our proposed method.

Assumptions and notations for this section.

A1 The point set Γ, is a realization of a collection of N i.i.d. random vectors $\{\boldsymbol{X}_i\}_{i=1}^N$, with each $\boldsymbol{X}_i \in \mathbb{R}^d$.
A2 The probability space induced by the random vector $\boldsymbol{X}_i$ is $(\mathbb{R}^d, \mathcal{F}, \mu)$, where $\mathcal{F}$ is the Borel σ-algebra on $\mathbb{R}^d$ and μ is a probability measure.
A3 The induced probability measure μ is also known as the distribution of $\boldsymbol{X}_i$, with compactly supported Lipschitz density ρ.
A4 The RaySense sketch uses the closest points feature space (3).
A5 In the case that $\boldsymbol{r}(s)$, for some s, has more than one nearest neighbor, we will randomly assign one.
A6 A ray, denoted by $\boldsymbol{r}(s)$, $0 \leqslant s \leqslant 1$, generated by the method introduced in Appendix A, is given in the embedding space $\mathbb{R}^d$ of Γ, and the support of ρ is centered.
A7 We assume $\text{supp}(\rho)$ can be covered by a finite union of hypercubes $\{\Omega_j\}_j$ in $\mathbb{R}^d$, each of non-zero probability measure, i.e., $\text{supp}(\rho) \subset \bigcup_j \Omega_j$, with $P_{\Omega_j} = \int_{x \in \Omega_j} \rho(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} > 0$, and $\{\Omega_j\}_j$ overlap with each other only on sets of measure zero.

Note that by A2–A3, we may regard ρ as representing the density of a solid body in $\mathbb{R}^d$. In other words, Γ does not consist of samples from a lower dimensional set. (RaySense can sample much more general sets, but the line integral analysis (Sect. 3.1.3) and the argument of sampling convex hull (Sect. 3.2.1) of this section may not hold.)

Much of our analysis is based on the *Voronoi cells* associated with each point in the point cloud.

Definition 5 The Voronoi cell of $\boldsymbol{x} \in \Gamma$ is defined as

$$V(\boldsymbol{x}) := \{\boldsymbol{y} \in \mathbb{R}^d : \mathcal{P}_\Gamma \boldsymbol{y} = \boldsymbol{x}\}. \tag{13}$$

In practice, we often sample rays $\boldsymbol{r} \sim \mathcal{L}$ of finite length as in Fig. 3, i.e., $\{\boldsymbol{r}_i\}_{i=1}^m \subset B_R(0)$ for some R. Correspondingly, we define the *truncated Voronoi cell*:

$$V^R(\boldsymbol{x}) := \{\boldsymbol{y} \in B_R(0) : \mathcal{P}_\Gamma \boldsymbol{y} = \boldsymbol{x}\}. \tag{14}$$

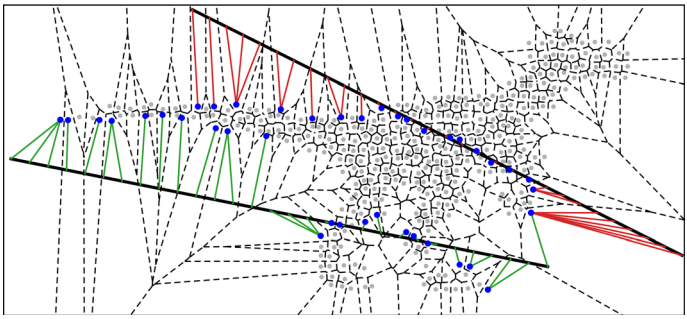


Fig. 3 A simple 2D point set (gray). Two rays (black) sense nearest neighbors of the point set (blue). Singular points, such as the tip of the tail, have larger Voronoi cells (dashed lines) and are more likely to be sampled. Closest point pairings are shown in green and red

3.1 Properties of Sampling with a Single Ray

3.1.1 Sampling Points with Larger Voronoi Cells

For discrete point sets, the likelihood that a ray senses a particular point is closely linked to the size of the Voronoi cell of the point. This relationship arises from the practice of utilizing closest-point sampling, which governs the selection of points by a given ray. In this regard, the Voronoi cell of a point in the point clouds is a fundamental geometric construct that plays a key role in determining the probability of detection. This observation is visually depicted in Fig. 3, which demonstrates that points having larger Voronoi cells are more likely to be detected by a given ray.

We refer to points with relatively large Voronoi cells as the "salient points" of Γ. When the probability density for Γ is compactly supported, the salient points of Γ tend to be situated in close proximity to geometric singularities present on the boundary of the support of the density; see Fig. 4 for a demonstration. This saliency characteristic will be further elaborated on in Sect. 3.2.1, where we will demonstrate how it manifests as the biased subsampling property of the RaySense method.

Furthermore, connections between the probability of a point $\in \Gamma$ being sampled and the size of its Voronoi cell can be made explicit in the following derivation.

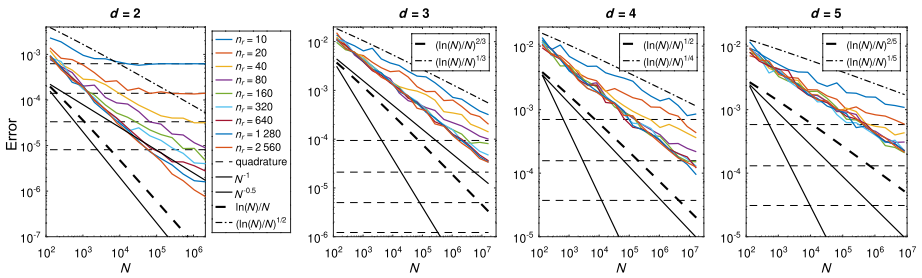


Fig. 4 Convergence studies for line integrals approximating from the RaySense sketch on point clouds sampled from a uniform density, in dimensions $d = 2, 3, 4, 5$. Horizontal dashed lines indicate error inherent to the trapezoidal rule quadrature schemes. Diagonal dashed lines indicate different convergence rates

Let $B_R(0) \in \mathbb{R}^d$ be a solid d-dimensional ball of radius R containing all the rays, and $\Gamma \subsetneq B_R(0)$ the finite point set containing N distinct points. Let $V_k := V(\boldsymbol{x}_k)$ denote the Voronoi cell for the kth point, $\boldsymbol{x}_k$ in Γ as in Definition 5. Let $\ell_k(\boldsymbol{r})$ denote the length of the segment of a ray, $\boldsymbol{r}$, that lies in the truncated Voronoi cell V_k^R. If $\boldsymbol{r}$ does not intersect V_k, $\ell_k(\boldsymbol{r}) := 0$, then $\ell_k(\boldsymbol{r})$ is a random variable indicating how much a ray $\boldsymbol{r}$ senses $\boldsymbol{x}_k \in \Gamma$, and we denote its expectation by $\mathbb{E}[\ell_k]$; in other words,

$$\mathbb{E}[\ell_k] := \int \ell_k(\omega) \mathrm{d}\mu_{\mathcal{L}}(\omega) = \mathbb{E}_{\boldsymbol{r}\sim\mathcal{L}}\Big[\int_r \chi_{V_k}\big(\boldsymbol{r}(s)\big)\mathrm{d}s\Big], \tag{15}$$

where $\mu_{\mathcal{L}}$ is the probability measure corresponding to the distribution of lines $\mathcal{L}$ introduced in Sect. 2, and χ_{V_k} is the indicator function of the Voronoi cell V_k. Consequently, the frequency of a ray $\boldsymbol{r}$ sampling a point $\boldsymbol{x}_k \in \Gamma$ is proportional to the d-dimensional Lebesgue measure of its Voronoi cell $V_k = V(\boldsymbol{x}_k)$.

3.1.2 Sampling Consistency

The Voronoi cell perspective provides a framework to analyze certain properties of RaySense. We will show in Theorem 1 that the sampling from a specific ray is consistent when the number of points N in the point clouds is large enough. We begin with some lemmas, with proofs appearing in Appendix C.

Our first lemma tells us how large N must be to have at least one sample in any region achieving a certain probability measure.

Lemma 1 *Suppose that* $\mathrm{supp}(\rho)$ *satisfies assumption A7. Let* Γ *be a set of* N *i.i.d. random samples drawn from* μ, *and* $p_0 \in (0,1)$. *If the number of sample points* $N > \nu\big(P\big)$ *where* $\nu : (0,1] \mapsto \mathbb{R}^+$ *is defined by*

$$\nu\big(P\big) := \frac{\sqrt{\ln\left(\frac{2}{1-p_0}\right)\left(\ln\left(\frac{2}{1-p_0}\right)+2P\right)}+P+\ln\left(\frac{2}{1-p_0}\right)}{P^2}, \tag{16}$$

then, with the probability greater than p_0, *at least one of the samples lies in every* Ω_j *with* $P_{\Omega_j} \geqslant P$. *Additionally, we note bounds for* $\nu(P)$:

$$\frac{2\ln\left(\frac{2}{1-p_0}\right)+P}{P^2} < \nu\big(P\big) < \frac{2\ln\left(\frac{2}{1-p_0}\right)+3P}{P^2}. \tag{17}$$

We notice that for any fixed $p_0 > 0$, $\nu(P) \sim \mathcal{O}\big(1/P^2\big)$ as $P \to 0$ indicating $\nu(P)$ is inversely proportional to P^2 asymptotically. This matches with the intuition that more points are needed to ensure sampling in regions with a smaller probability measure.

The next two lemmas reveal that the volume of the Voronoi cell for a sample point amongst the others in the point cloud Γ decreases to zero with high probability as the number of sampled points tends to ∞.

Lemma 2 *Suppose* ρ *is L-Lipschitz and* $\mathrm{supp}(\rho)$ *satisfies assumption A7. Given* Γ *a set of* N *i.i.d. random samples drawn from* μ, *for* N *large enough, the size of the Voronoi cell of a*

sample point $\boldsymbol{x} \in \Gamma$ *in the interior of* $\mathrm{supp}(\rho)$ *is inversely proportional to its local density* $\rho(\boldsymbol{x})$, *and with probability at least* p_0 *its diameter has the following upper bound*:

$$\mathrm{diam}(V(\boldsymbol{x})) \leqslant 3\sqrt{d}\left(\frac{21+7\left(9+8N\ln\left(\frac{2}{1-p_0}\right)\right)^{\frac{1}{2}}}{6\rho(\boldsymbol{x})N}\right)^{\frac{1}{d}}.$$

When the underlying distribution μ is uniform, $\rho(\boldsymbol{x})$ is the same everywhere inside $\mathrm{supp}(\rho)$, therefore the Voronoi diameter for every $\boldsymbol{x}$ in the interior should shrink uniformly. However, a better bound can be obtained for this case, as shown in the following lemma.

Lemma 3 *If* μ *is a uniform distribution*, *then given* Γ *with* N *large enough*, *with probability at least* p_0, *the diameter of the Voronoi cell of any sample point in the interior has the bound*

$$\mathrm{diam}(V) \leqslant 3\sqrt{d}\left(\left\lfloor\frac{1}{c(N)N}\ln\frac{N}{1-p_0}\right\rfloor\right)^{\frac{1}{d}}$$

with some $c(N)$ *such that* $c(N) \to 1$ *as* $N \to \infty$.

Theorem 1 (Consistency of sampling) *Under A*1–*A*7, *suppose* Γ_1 *and* Γ_2 *are two point clouds sampled from the same distribution*, *with* N_1 *and* N_2 *points respectively*, *where in general* $N_1 \neq N_2$. *Assume further that* $\mathrm{supp}(\rho)$ *is convex. For a ray* $\boldsymbol{r}(s)$ *using* n_r *uniformly-spaced discrete points to sample*, *for* $N = \min(N_1, N_2)$ *sufficiently large*, *the RaySense sketches in the closest point feature space* $S[\Gamma_1]$ *and* $S[\Gamma_2]$ *satisfy*

$$\left\|S[\Gamma_1] - S[\Gamma_2]\right\|_{\mathrm{F}} \leqslant \varepsilon(N), \tag{18}$$

where $\varepsilon > 0$ *and* $\varepsilon \to 0$ *as* $N \to \infty$.

Remark 1 The assumption that $\mathrm{supp}(\rho)$ is convex is stronger than needed in many cases; it excludes the situation where some point $\boldsymbol{r}_{i,j}$ in the discretized ray set is equidistant to two or more points on the non-convex $\mathrm{supp}(\rho)$ that are widely separated, which could lead to an unstable assignment of nearest neighbors. However, in practical scenarios where a ray is chosen randomly, this situation is unlikely to occur. Further details can be found at the end of Appendix C.

The consistency of sampling ensures the RaySense data on a specific ray would be close when sampling the same object, therefore one can expect a similar property for the RaySense sketch tensor where multiple rays are used, which will be discussed in Sect. 3.2.1.

3.1.3 Approximate Line Integrals

We demonstrate that the RaySense approach enables the computation of line integrals of functions defined on the points in a point cloud. Suppose we have a point cloud Γ representing an object in $\mathbb{R}^d$. As N increases, the point cloud becomes denser, and for any $\boldsymbol{r}(s)$

lies inside $\mathrm{supp}(\rho)$, $\mathcal{P}_\Gamma \boldsymbol{r}(s) \approx \boldsymbol{r}(s)$ as the Voronoi cells shrink around each point in Γ. If we have a smooth function $g : \mathbb{R}^d \mapsto \mathbb{R}$ evaluated on the point cloud, then $g(\mathcal{P}_\Gamma \boldsymbol{r}(s)) \approx g(\boldsymbol{r}(s))$ and we expect that integrals of g along lines can be approximated by integrals of the RaySense sketch $\mathcal{S}[\Gamma, g]$ introduced in Sect. 2.2.2 (and quadrature of the discrete RaySense sketch $\mathcal{S}_{m,\delta r}[\Gamma, g]$).

The following shows that the RaySense integral is an approximation to the line integral along $\boldsymbol{r}(s)$ provided the point cloud is dense enough.

Theorem 2 *Suppose that $g \in C(\mathbb{R}^d; \mathbb{R})$ is J-Lipschitz, ray $\boldsymbol{r}(s) \in \mathrm{supp}(\rho)$ for $0 \leqslant s \leqslant 1$, and Γ is a set of N i.i.d. random samples drawn from μ, with corresponding RaySense sketch $\mathcal{S}[\Gamma, g]$, then the difference between the RaySense integral of g and the line integral of g has the following bound*:

$$\left| \int_0^1 g\big(\boldsymbol{r}(s)\big) \mathrm{d}s - \int_0^1 g\big(\mathcal{P}_\Gamma \boldsymbol{r}(s)\big) \mathrm{d}s \right| \leqslant J \sum_{\boldsymbol{x} \in \Gamma} \int_0^1 \chi_{V(\boldsymbol{x})}\big(\boldsymbol{r}(s)\big) \|\boldsymbol{r}(s) - \mathcal{P}_\Gamma \boldsymbol{r}(s)\| \mathrm{d}s, \quad (19)$$

where $\chi_{V(\boldsymbol{x})}$ is the indicator function of the Voronoi cell $V(\boldsymbol{x})$ for $\boldsymbol{x} \in \Gamma$.

Proof For a fixed number of sampling points N, the approximation error is given by

$$\begin{aligned} \left| \int_0^1 g\big(\boldsymbol{r}(s)\big) \mathrm{d}s - \int_0^1 g\big(\mathcal{P}_\Gamma \boldsymbol{r}(s)\big) \mathrm{d}s \right| &\leqslant \int_0^1 \left| g\big(\boldsymbol{r}(s)\big) - g\big(\mathcal{P}_\Gamma \boldsymbol{r}(s)\big) \right| \mathrm{d}s \\ &= \sum_{\boldsymbol{x} \in \Gamma} \int_0^1 \chi_{V(\boldsymbol{x})}\big(\boldsymbol{r}(s)\big) \left| g\big(\boldsymbol{r}(s)\big) - g\big(\mathcal{P}_\Gamma \boldsymbol{r}(s)\big) \right| \mathrm{d}s \\ &\leqslant J \sum_{\boldsymbol{x} \in \Gamma} \int_0^1 \chi_{V(\boldsymbol{x})}\big(\boldsymbol{r}(s)\big) \|\boldsymbol{r}(s) - \mathcal{P}_\Gamma \boldsymbol{r}(s)\| \mathrm{d}s. \end{aligned}$$

In scattered data interpolation, the nearest neighbor interpolation would have an error of $\mathcal{O}(h)$ where $h = \max_k \mathrm{diam}(V_k)$. Integration of the interpolated data would result in an error of $\mathcal{O}(h^2)$, consistent with Theorem 2.

Intuitively, we expect the RaySense line integral to converge in the limit of $N \to \infty$. Here we show the corresponding convergence result for the case of uniform density from the perspective of a Poisson point process [9]. Details of the proof are given in Appendix D.

Theorem 3 *Suppose $g \in C(\mathbb{R}^d; \mathbb{R})$ is J-Lipschitz, ρ is uniform and satisfies assumption A7, and the ray $\boldsymbol{r}(s) \in \mathrm{supp}(\rho)$ for $0 \leqslant s \leqslant 1$, then given Γ a set of N i.i.d. random samples as a realization of a Poisson point process with the corresponding RaySense sketch $\mathcal{S}[\Gamma, g]$ for the ray, the probability that the following holds tends to 1 as $N \to \infty$:*

$$\left| \int_0^1 g\big(\boldsymbol{r}(s)\big) \mathrm{d}s - \int_0^1 g\big(\mathcal{P}_\Gamma \boldsymbol{r}(s)\big) \mathrm{d}s \right| \leqslant c(d, J) N^{-\frac{1}{d} + \varepsilon(d+1)}$$

for any small $\varepsilon > 0$.

When $\varepsilon < \frac{1}{(d+1)^2}$, the line integral error converges to 0 as $N \to \infty$ in a rate $\mathcal{O}(N^{-\frac{1}{d}})$.

We first confirm this rate with numerical experiments. We return to explore applications of integral transforms in Sect. 4.3. Figure 4 shows convergence studies of the RaySense integrals for the uniform density case from Theorem 3. The 5-dimensional (5D) example uses an integrand of $g(x_1, x_2, x_3, x_4, x_5) = \cos(x_1 x_2) - x_4 x_5 \sin(x_3)$ and a line $\boldsymbol{r}(s) = \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} s + \frac{\langle 1,1,1,1,1\rangle}{10}$ with $\boldsymbol{v} = \langle 2, 3, 4, 5, 6\rangle$. In four, three, and two dimensions, we use $g(x_1, x_2, x_3, x_4, 1)$, $g(x, y, z, 1, 1)$, and $g(x, y, 0, 0, 0)$, respectively, and drop unneeded components from the line. The exact line integrals were computed with the Octave Symbolic package [34] which uses SymPy [37] and mpmath [56]. Each experiment is averaged over 50 runs.

In Fig. 4, we see that for each fixed number of sample points n_r along the ray, the error decreases at the rate discussed above. From the factor of two in the distance between the results of each fixed n_r, infer a first-order decrease in n_r, and taken together an overall faster rate of convergence if both N and n_r are increased. In all cases in Fig. 4, the experimental convergence rate is bounded by $\mathcal{O}\left(\frac{1}{N}\ln N\right)^{\frac{1}{d}}$ which is asymptotically close to the predicted rate given in Theorem 3 since ε can be taken arbitrarily small. However, when n_r is large, we appear to achieve a faster rate of $\mathcal{O}\left(\frac{1}{N}\ln N\right)^{\frac{2}{d}}$. This suggests a tighter analysis may be possible in the future.

3.1.4 Ray not Fully Contained in supp(ρ)

For any portion of $\boldsymbol{r}(s)$ that lies outside of supp(ρ) such portion should take no values when computing the line integral if g is only defined on supp(ρ). Thus, the post-processing procedure involves eliminating sampling points on $\boldsymbol{r}(s)$ that are outside of supp(ρ). When N is large, this can be accomplished by augmenting the RaySense feature space with vectors to the closest point as in (9) rather than using A4, and redefining $g(\mathcal{P}_\Gamma \boldsymbol{r}(s)) = 0$ if the distance to the closest point of $\boldsymbol{r}(s)$ is beyond some small threshold.

3.2 Properties of Sampling with Multiple Rays

When applying RaySense to the point set $\Gamma \in \mathbb{R}^d$ with m rays and n_r points on each ray, the RaySense sketch $S(\Gamma)$ is an $m \times n_r \times d$ tensor defined in (10a). In the following sections, we investigate properties of the RaySense sketch as an extension of the properties of RaySense using a single ray previously discussed.

3.2.1 Biased Samplings Toward Salient Points

A direct consequence from Sect. 3.1.1 is that the RaySense sketch $S(\Gamma)$ tends to repeatedly sample points with larger Voronoi cells, indicating a bias toward these salient points. Figure 5 demonstrates this property by visually presenting the frequency of the sampled points by the size of the plotted blue dots.

Another notable feature from Fig. 5 is that the biased subsample generated by RaySense also depicts the outline of the object. This is because points with larger Voronoi cells usually situate near the boundary and regions with positive curvatures. The following proposition gives a sufficient condition to identify such points.

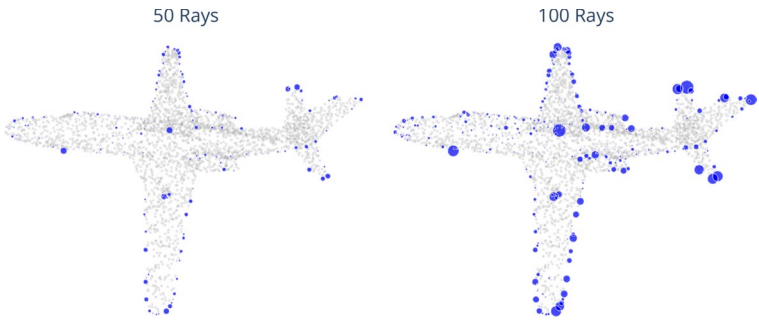


Fig. 5 Points sampled by RaySense using different numbers of rays are indicated as blue dots. The larger blue dots correspond to points that are more frequently sampled. The effect of sampling saliency becomes more apparent as the number of rays increases. Each ray contains 30 sample points

Proposition 1 *Vertices of the convex hull of the point cloud Γ will be frequently sampled by RaySense, when using sufficiently long rays.*

Proof Because supp(ρ) is compact, the Voronoi cell for vertices on the convex hull is unbounded. Therefore, one can make the volumes of their truncated Voronoi cell (14), as large as desired using rays with suitable length. And recall from Sect. 3.1.1 that the frequency of being sampled is closely related to the measures of the Voronoi cell.

Remark 2 The arguments in the proof also imply when the length of the line segment $\to \infty$, with a high probability RaySense is sampling mostly the convex hull of the point sets when supp(ρ) is compact. This can be viewed as a different approach to approximate convex hulls using rays and curvatures from [16].

Proposition 1 applies to abstract datasets as well; we consider the MNIST dataset [29], treating each image as a point in $d = 784$ dimensions. Here Γ is the point set consisting of all images of the same digit. Figure 6 shows the average digits over the whole dataset, versus the average of those sampled by RaySense.

In the context of MNIST, salient points are digits that are drawn using less typical strokes (according to the data). These are the data points that may be harder to classify, since they appear less frequently in the data. RaySense may be used to determine the most *useful* data points to label, as in active learning [51]. RaySense also provides a special notion of importance sampling based on the notion of saliency described above. An application of such a property is further discussed in Sect. 4.2.

Fig. 6 Each digit averaged over the entire data set (top) versus those sampled by RaySense (bottom)

3.2.2 Invariant Histogram

The relationship between the frequency of a point in Γ being sampled and the measure of its Voronoi cell derived in Sect. 3.1.1 also provides a crucial guarantee. It ensures the RaySense histogram introduced in (11) possesses a well-defined limit as the number of rays $m \to \infty$. This convergence can be better elucidated using an alternative formulation of H_k from (11), which incorporates the concept of Voronoi cell.

Draw m rays, $\boldsymbol{r}_1, \boldsymbol{r}_2, \cdots, \boldsymbol{r}_m$, from the distribution $\mathcal{L}$ that are contained in some $B_R(0)$ of suitable R. Enumerate this set of points by $\boldsymbol{r}_{i,j}$, and the spacing between two adjacent points is given by δr. The closest point of $\boldsymbol{r}_{i,j}$ is $\boldsymbol{x}_k$ if $\boldsymbol{r}_{i,j} \in V_k := V(\boldsymbol{x}_k)$. Therefore, the bin value of the RaySense histogram can also be given as

$$H_k = H\big(\boldsymbol{x}_k; \mathcal{S}_{m,\delta r}(\Gamma)\big) := \sum_{i=1}^{m} \sum_{\boldsymbol{r}_{i,j} \in V_k} 1,$$

where $\mathcal{S}_{m,\delta r}(\Gamma)$ denotes the closest point sketch tensor using m rays and δr spacing. The following theorem shows that under proper normalization, H_k can be thought as a hybrid Monte-Carlo approximation to $\mathbb{E}[\ell_k]$ defined in (15).

Theorem 4 *The normalized bin values $\tilde{H}_k$ with the normalized constant $\delta \frac{r}{m}$, i.e.,*

$$\tilde{H}_k := \frac{\delta r}{m} H\big(\boldsymbol{x}_k; \mathcal{S}_{m,\delta r}(\Gamma)\big) = \frac{1}{m} \sum_{i=1}^{m} \sum_{\boldsymbol{r}_{i,j} \in V_k} \delta r$$

have the limit

$$\lim_{\substack{m \to \infty \\ \delta r \to 0}} \tilde{H}_k(\mathcal{S}_{m,\delta r}(\Gamma)) = \mathbb{E}[\ell_k].$$

Proof

$$\begin{aligned} \lim_{\substack{m \to \infty \\ \delta r \to 0}} \tilde{H}_k(\mathcal{S}_{m,\delta r}(\Gamma)) &= \lim_{\substack{m \to \infty \\ \delta r \to 0}} \frac{1}{m} \sum_{i=1}^{m} \sum_{\boldsymbol{r}_{i,j} \in V_k} \delta r = \lim_{\delta r \to 0} \mathbb{E}_{\boldsymbol{r}_i \sim \mathcal{L}} \Big[\sum_{\boldsymbol{r}_{i,j} \in V_k} \delta r \Big] \\ &= \mathbb{E}_{\boldsymbol{r}_i \sim \mathcal{L}} \Big[\lim_{\delta r \to 0} \sum_{\boldsymbol{r}_{i,j} \in V_k} \delta r \Big] = \mathbb{E}_{\boldsymbol{r}_i \sim \mathcal{L}} \Big[\int_{\boldsymbol{r}_i} \chi_{V_k}\big(\boldsymbol{r}(s)\big) \mathrm{d}s \Big] = \mathbb{E}[\ell_k], \end{aligned}$$

where χ_{V_k} is the indicator function of V_k. Interchanging order of the limit and the expectation follows from the dominated convergence theorem since $\sum_{\boldsymbol{r}_{i,j} \in V_k} \delta r < \chi_{V_k}\big(\boldsymbol{r}(s)\big) + 2\delta r$.

Monte-Carlo approximations of integrals converge with a rate independent of the dimension [4]. Consequently, for sufficiently many randomly selected rays, the histogram is essentially independent of the rays that are actually used.

Similar arguments show that the sampling of any function of the data set will be independent of the actual ray set, since the histograms are identical in the limit. More precisely, suppose $g : \boldsymbol{x} \in \Gamma \mapsto \mathbb{R}$ is some finite function, then

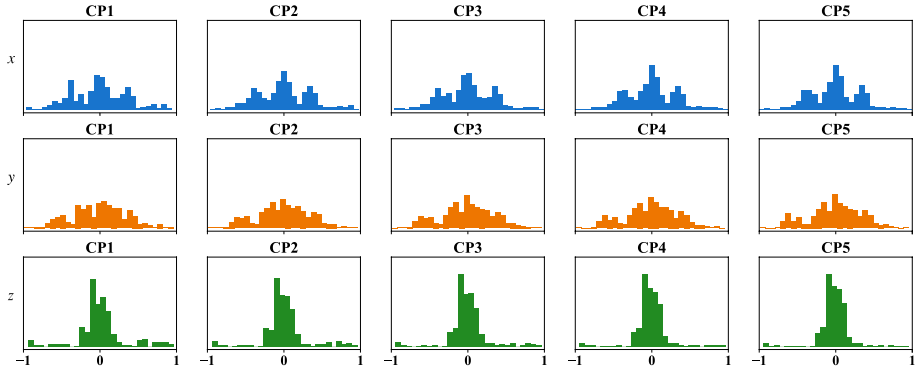


Fig. 7 Histogram of the $\eta = 5$ nearest neighbors sampled by RaySense, where the underlying point cloud is polluted by 50 outliers uniformly sampled from the unit ball. Rows correspond to different coordinates while columns correspond to different closest points. Outliers introduce extreme values on coordinates of CP1 but these effects are significantly mitigated on CP5

$$\lim_{\substack{m\to\infty\\ \delta r\to 0}} \frac{1}{m}\sum_{i=1}^{m}\sum_{\boldsymbol{r}_{i,j}\in V_k} g(\boldsymbol{x}_k)\delta r = \mathbb{E}[g(\boldsymbol{x}_k)\ell_k].$$

In Fig. 2, we show the histograms of the coordinates of the RaySensed points of Γ.

The integral $\mathbb{E}[\ell_k]$ (or $\mathbb{E}[g(\boldsymbol{x}_k)\ell_k]$ for continuous g) depends smoothly on Γ, and is therefore stable against perturbation to the coordinates of the points in Γ. However, the effect of introducing new members to Γ, such as outliers, will be non-negligible. One possible way to overcome this is to use multiple nearest neighbors for points on the rays. In Fig. 7, we show coordinates of the $\eta = 5$ nearest neighbors sampled by RaySense under the presence of outliers. It is observed that the features of the ηth nearest neighbor for η large is more robust against outliers while also maintaining the desired histogram information. In Sect. 4.5, we will also demonstrate the effectiveness of this idea in dealing with outliers in practical tasks.

By considering a similar argument as in the proof of Theorem 1, we can provide a simple and intuitive explanation for the robustness of outliers when using more nearest neighbors: when the underlying point cloud is dense enough, if an outlier disrupts the nearest neighbor search, excluding the outlier and finding the next few nearest neighbors would mitigate the impact caused by the outlier; if an outlier does not dominate the nearest neighbor search, then the next few nearest neighbors with high probability would also originate from a small neighborhood centered around the first nearest neighbor. Therefore, increasing the number of nearest neighbors enhances the stability of RaySense.

4 Examples of Applications

4.1 Comparison of Histograms of RaySense Samples

We experiment by comparing Γ drawn from 16 384 objects of 16 categories from the ShapeNet dataset [5]. Let β^i be the label for object Γ_i. We compute the histograms h_x^i, h_y^i, h_z^i of the x, y, z coordinates, respectively, for points sampled by 50 rays with $n_r = 10$ samples

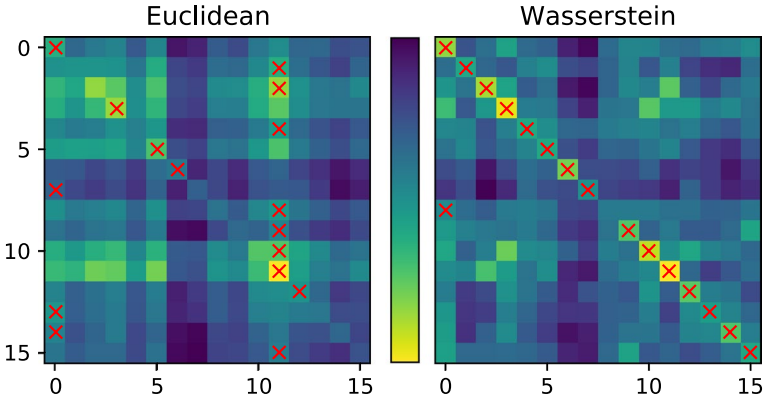


Fig. 8 Comparison of histograms of the x, y, z coordinates of points sampled by RaySense, using ℓ^2 and the Wasserstein distance W_1. Rows and columns correspond to object labels. Red × indicates the location of the argmin along each row

per ray. We compare the histograms against those corresponding to other objects in the dataset, using

$$D_{i,j} = d(h_x^i, h_x^j) + d(h_y^i, h_y^j) + d(h_z^i, h_z^j),$$

where $d(\cdot,\cdot)$ is either the ℓ_2 or Wasserstein-1 distance. We sum D according to the respective labels

$$M_{a,b} \propto \sum_{i:\beta^i=a} \sum_{j:\beta^j=b} D_{i,j}, \qquad a, b = 1, \cdots, 16,$$

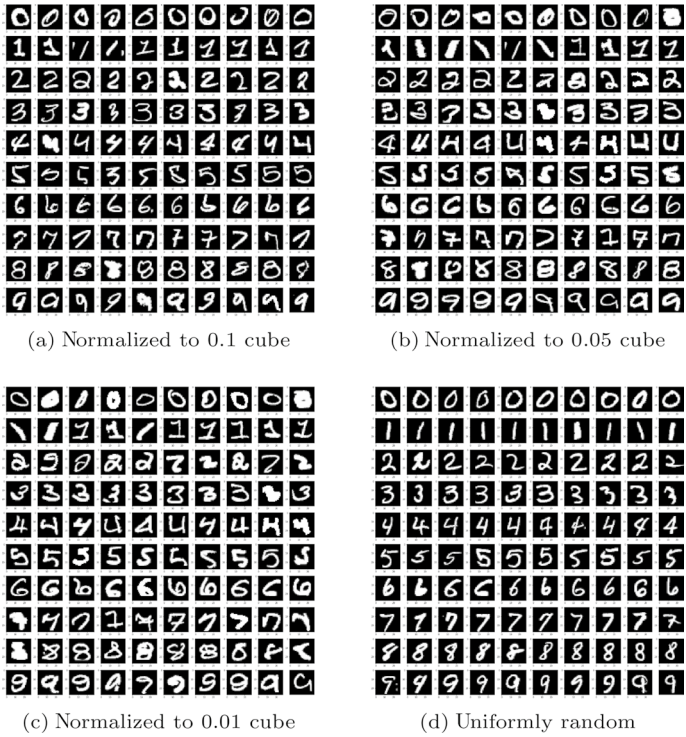
and normalize by the number of occurrences for each a, b pair. Figure 8 shows the matrix of pairwise distances M between the 16 object categories.

Ideally, intra-object distances would be small, while inter-object distances would be large. As expected, Wasserstein-1 is a better metric for comparing histograms. Still, not all objects are correctly classified. When comparing histograms is not sufficient, we consider using neural networks to learn more complex mappings, such as in Sect. 4.5.

4.2 Salient Points in the MNIST Data

From previous discussion and simulation in Sects. 3.1.1 and 3.2.1, we know RaySense has the ability to detect salient points or boundary points. Here we provide further visualization of RaySense salient points on the MNIST dataset.

By vectorizing the MNIST image, each image is a vector in $\mathbb{R}^{784}$ with pixel value from 0 to 255. We generate the random ray set in this ambient space using Method R1 in Appendix A, where each ray has the fixed-length 1, with centers uniformly shifted in the half cube $[-\frac{1}{2}, \frac{1}{2}]^{784}$. Each ray set has $m = 256$ random rays, with $n_r = 64$ equi-spaced points on each ray. To ensure a good coverage over the data manifold, we rescale the MNIST image by entry-wise dividing so that each data point is constrained in an ℓ^∞ ball of a certain radius as introduced below; we also shift the dataset to have a mean 0.



(a) Normalized to 0.1 cube (b) Normalized to 0.05 cube

(c) Normalized to 0.01 cube (d) Uniformly random

Fig. 9 MNIST digit images with the highest RaySense sampling frequencies for each class. Three different normalizations are shown in **(a)**–**(c)**. Compared to a uniformly random subsample (**d**), we see a wider variety of hand-writing styles in the RaySense output

As mentioned, the RaySense salient points are those in Γ sampled most frequently by points from the rays. We record the sampling frequency for each MNIST image, and in Fig. 9 we plot the top-10 images with highest frequency for each class. From the figure, we see that the salient points often correspond to digits with untypical strokes and writing styles, similar to the conclusion obtained from Fig. 6. Figure 9 further shows that different normalizations of the data (by using scaling values 2 550, 5 100, and 25 500) also affect the sampling. This phenomenon can be better understood from the perspective of truncated Voronoi cell (14): when the scale of the point clouds shrinks while the length of rays remains constant, it has a similar effect as increasing the length of the rays while keeping the point clouds unchanged, causing the truncated Voronoi cells to grow. Specifically, the truncated Voronoi cells associated with salient points exhibit a larger growth rate, as their Voronoi cells are typically unbounded, e.g., Proposition 1, making the subsampling even more biased when normalized to a smaller cube.

4.3 RaySense and Integral Transforms

A line $\boldsymbol{r}$ in $\mathbb{R}^d$ in the direction of $\boldsymbol{\theta} \in \mathbb{S}^{d-1}$ has parameterization $\boldsymbol{r}(s) = \boldsymbol{b} + s\boldsymbol{\theta}$, $s \in (-\infty, \infty)$, with $\boldsymbol{b} \in \mathbb{R}^d$ a reference point on the line. Without loss of generality, let $\boldsymbol{b}$ be in $\boldsymbol{\theta}^{\perp}$, which is a

hyperplane orthogonal to $\boldsymbol{\theta}$ passing through the origin. The X-ray transform for a non-negative continuous function g with compact support [39] is defined on the set of all lines in $\mathbb{R}^d$ by

$$\mathcal{X}[g](\boldsymbol{b},\boldsymbol{\theta}) := \int_{-\infty}^{\infty} g(\boldsymbol{b}+s\boldsymbol{\theta})\mathrm{d}s. \tag{20}$$

The spectrum of g can be obtained via the Fourier slice theorem [55]:

$$\mathcal{F}[\mathcal{X}g](\boldsymbol{\theta},\boldsymbol{\xi}) = \mathcal{F}[g](\boldsymbol{\xi}), \quad \boldsymbol{\xi}\in\boldsymbol{\theta}^{\perp}.$$

When we restrict $\boldsymbol{\xi}$ to be only on a line in $\boldsymbol{\theta}^{\perp}$, we are effectively collecting information on a 2D slice of g parallel to $\boldsymbol{\theta}^{\perp}$.

However, when the function g only has a sampling representation, e.g., a point cloud, it is non-trivial to compute such integrals. In Sect. 3.1.3, we showed that if $\boldsymbol{r}$ is a member of the sampling ray set, one can compute an approximation of (20) from the RaySense sketch obtained from $\{\boldsymbol{x}_k, g(\boldsymbol{x}_k)\}_{k=1}^N$, where $\{\boldsymbol{x}_k\}$ are i.i.d. samples from a known probability density ρ. Thus, RaySense provides a convenient alternative in obtaining (or, in a sense, defining) the Fourier slices of the discrete data set $\{\boldsymbol{x}_k, g(\boldsymbol{x}_k)\}_{k=1}^N$. Since (20) is defined for any dimension, one can approximate the X-ray transform from a RaySense sketch using suitable ray sets, or, in the random case, RaySense integrals can be regarded as randomized approximations of X-ray transforms.

In Fig. 10 we show an example of using RaySense sampling with prescribed (rather than random) rays to approximate the Radon transform. In this experiment, a point cloud Γ (Fig. 10a top) with 15 010 points, is sampled from density $\rho = \frac{1}{2} - 3x\mathrm{e}^{-9x^2-9y^2}$ (Fig. 10a bottom)—note denser (darker) region on left and sparser (lighter) region on right. Γ has data shown in Fig. 10b evaluated from the piecewise constant function g, shown by the solid colours (for visualization only; g is only known at the discrete points in Γ). Blue lines show the locations of the RaySense sketch for one particular angle (illustrated with 21 rays but the computation uses 100). We note increasingly jagged lines to the right where the point cloud is sparser. Figure 10c shows that approximate Radon transform computed over 180° in steps of one degree by integrating the RaySense sketch using trapezoidal rule at $n_r = 64$ points per ray. Figure 10d shows the filtered back projection computed by the Octave Image package [12]. Note a more jagged reconstruction on the right where the point cloud is sparsest. If we instead used random rays, we could generate samples at scattered points in the sinogram (Fig. 10c) which could then be used for an approximate inverse transform.

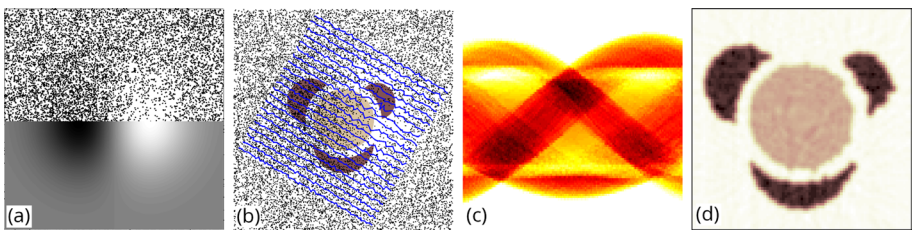


Fig. 10 Approximate Radon transform computed with RaySense from point cloud data (**a**–**c**) and filtered back projection reconstruction (**d**)

4.4 Point Cloud Registration

In this section, we explore the application of RaySense to the point cloud registration problem. Given two point sets Γ and $\tilde{\Gamma}$ in three dimensions consisting of distinct points, registration aims to find the 3D rotation matrix $\boldsymbol{U}$ and translation vector $\boldsymbol{b}$ to minimize the Euclidean norm of points in correspondence. When the correspondence is known, this is the orthogonal Procrustes problem and the solution can be obtained explicitly via the singular value decomposition. When the correspondence is unknown, one can formulate an optimization problem and solve it with various carefully-designed algorithms. Here we choose to use the Iterative Closest Point (ICP) [3] due to its simplicity, which minimizes point-wise Euclidean distance iteratively from the optimization problem

$$\min_{\boldsymbol{U}\in SO(3),\boldsymbol{b}\in\mathbb{R}^3} \sum_{\boldsymbol{x}_j\in\tilde{\Gamma}} \min_{\boldsymbol{y}\in\Gamma} \|U(\boldsymbol{x}_j+\boldsymbol{b})-\boldsymbol{y}\|_2^2.$$

We set up the problem using the Stanford Dragon [7] as a point cloud Γ with 100 000 points. We artificially generate the target point cloud to register by rotating by $\frac{\pi}{3}$ in one direction. We compare the performance of ICP in three scenarios: (i) the original dense point clouds, (ii) a uniformly random subsampling (in index) of the point clouds, (iii) RaySense closest point samples using $\mathcal{S}[\Gamma]$ (without repetition of sampled points) of each point cloud. Specifically, we use $m = 512$ rays, each with $n_r = 64$ sample points, to subsample the original point cloud in RaySense, which usually generates a set of around 800 unique points. We then sample the second point cloud with a different set of rays. For fair comparison, we also subsample 800 points in the case of uniformly random subsampling. We use the root mean square error (RMSE) as our metric, and we also record the convergence time, where the convergence criteria is a threshold of the relative RMSE. We summarize the performance results in Table 1, and we provide some visualization to compare the three different settings in Fig. 11.

From Table 1, it is clear that both the sampling schemes accelerate the registration process drastically by considering only a portion of the original dense point cloud. It also suggests that RaySense sample has a slight advantage over the uniform random sample, in both accuracy and convergence time. However, generating the RaySense samples on the fly needs around 0.65 s on average, while generating a random subsample requires only 0.01 s.

Figure 11c again shows that RaySense is sampling salient features. Thus a possible improvement is to use the repetition information from RaySense sampling to perform a weighted registration, for example with the (autodetected) salient features receiving higher weights. This is left for future investigation.

Table 1 Sample point cloud registration result. Performances are evaluated by registration accuracy (measured by root mean squared error (RMSE)) and computation times. The statistics reported are averaged over 5 runs. "RMSE" is evaluated over the subsampled points while "RMSE(full)" is evaluated over the original point cloud

	Number of points	RMSE	RMSE (full)	Convergence time (s)
Vanilla ICP	100 000	4.544E−06	4.544E−06	6.319
ICP + random	800	4.509E−02	3.053E−03	0.019 2
ICP + RaySense	804.6	2.601E−02	1.077E−03	0.011 6

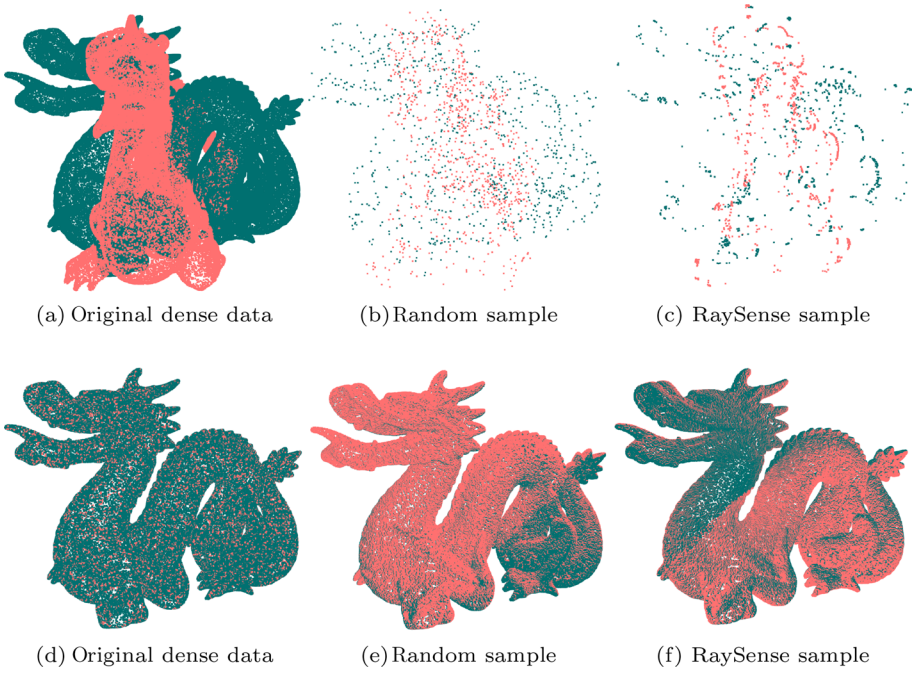


Fig. 11 Registration simulation on a rotated Stanford dragon. Top row: initial pose and sparse samples; bottom row: registration results. Note that the visualizations (**e**) and (**f**) are obtained by applying the transformation computed from the sparse samples in (**b**) and (**c**) to the original dense point clouds

4.5 Point Cloud Classification Using Neural Networks

We use the RaySense sketch to classify objects from the ModelNet dataset [62], using a neural network, which we call RayNN. RayNN can use features from different sampling operators $\mathcal{S}$ introduced in Sect. 2 as inputs. When using multiple nearest neighbors: $\mathcal{S}[\Gamma, [1, 2, \cdots, \eta]]$, we denote our models by RayNN-cpη. For our implementation, while

Table 2 ModelNet classification results. Here we report our best accuracy results over all experiments. For reference, the test scores for RayNN-cp5 ($m = 32$) has mean around 90.31% and standard deviation around 0.25% over 600 tests. The best score for each dataset is in bold

	ModelNet10	ModelNet40
PointNet [44]	–	89.2
PointNet++ [45]	–	90.7
ECC [53]	90.8	87.4
kd-net [27]	93.3	90.6
PointCNN [31]	–	92.5
PCNN [1]	94.9	92.3
DGCNN [61]	–	**92.9**
RayNN-cp1 ($m = 16$)	94.05	90.84
RayNN-cp5 ($m = 32$)	**95.04**	90.96

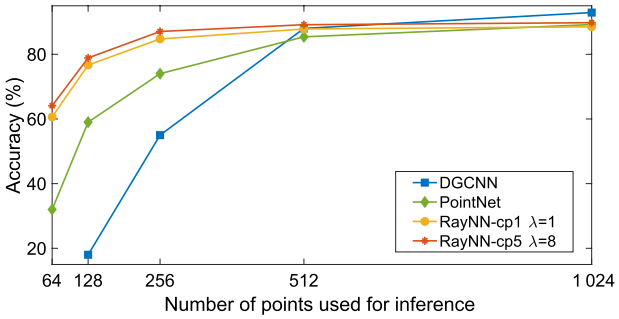


Fig. 12 Testing DGCNN [61], PointNet [44] and RayNN on ModelNet40 with missing data

Table 3 Accuracy when testing with a reduced ray set. RayNN-cp1 was trained using $m = 32$ rays. Results averaged over 5 runs

ModelNet40				
$\hat{m}$	32 (%)	16 (%)	8 (%)	4 (%)
$\lambda = 1$	88.50	86.13	74.64	43.28
$\lambda = 8$	89.77	88.94	82.97	55.24

Table 4 Outliers sampled uniformly from the unit sphere are introduced during testing. The networks are trained without any outliers. Results averaged over 5 runs. Best scores for each dataset are in bold

	No outliers (%)	5 outliers (%)	10 outliers (%)
ModelNet10			
RayNN-cp1	93.26	79.76	53.94
RayNN-cp5	**93.85**	**92.66**	**90.90**
PointNet.pytorch	91.08	48.57	25.55
ModelNet40			
RayNN-cp1	89.77	54.66	20.95
RayNN-cp5	**90.38**	**88.49**	**78.06**
PointNet.pytorch	87.15	34.05	17.48

we might use different numbers of nearest neighbors, we always include the closest point coordinates and the vector to closest points in our feature space $\mathbb{R}^c$ ($c \geqslant 6$ fixed). Details of the implementation can be found in Appendix B.

We compare with some well-known methods for 3D point cloud classification tasks. In addition to the results reported by [44], we also compare against PointNet.pytorch, a PyTorch re-implementation [63] of PointNet. In all our experiments, we report overall accuracy. Table 2 shows RayNN is competitive. To investigate the robustness of our network, we performed several more experiments.

Table 5 Top: storage and timings for RayNN-cp1 and PointNet.pytorch on ModelNet40 using one Nvidia 1 080-Ti GPU and batch size 32. The preprocessing and forward time are both measured per batch. Bottom: data from [61] is included only for reference; no proper basis for direct comparison

	Model size (MB)	Forward time (ms)	Preprocessing time (ms)	Time per epoch (s)
PointNet.pytorch	14	12	**3.6**	**14**
RayNN-cp1	**4.5**	**2**	7.5	22
PointNet [44]	40	16.6	–	–
PCNN [1]	94	117	–	–
DGCNN [61]	21	27.2	–	–

Robustness to sample size We repeat the experiments in [44, 61] whereby, after training, data is randomly removed prior to testing on the remaining points. The results in Fig. 12 show that RayNN performs very well with significant missing data.

Using fewer rays We experiment with training using a full set of $m = 32$ rays but test using smaller number $\hat{m}$ of rays. Table 3 shows that RayNN can achieve a reasonable score even if only $\hat{m} = 4$ rays are used for inference.

Robustness to outliers This experiment simulates situations where noise severely perturbs the original data during testing. We compare the performance of RayNN-cp1, RayNN-cp5, and PointNet.pytorch in Table 4. The comparison reveals RaySense's capability in handling unexpected outliers, especially when additional nearest neighbors are used. Note the experiment here is different from that in [44] where the outliers are fixed and included in the training set.

Comparison of model complexity

Table 5 shows that our network has an advantage in model size and feed-forward time even against the simple and efficient PointNet. In both training and testing, there is some overhead in data preprocessing to build a kd-tree, generate rays, and perform the nearest-neighbor queries to form the RaySense sketch. For point clouds of around $N = 1\,024$, these costs are not too onerous in practice as shown in Table 5.

The convolution layers have $48c + 840\,016$ parameters, where c is the dimension of input feature space. The fully-connected layers have $64K + 278\,528$ parameters, where K is the number of output classes. In total, our network has $1.1 \times 10^6 + 48c + 64K \approx 1.1$ M parameters. In comparison, PointNet [44] contains 3.5 M parameters.

5 Summary

RaySense is a sampling technique based on projecting random rays onto a data set. This projection involves finding the nearest neighbors in the data for points on the rays. These nearest neighbors collectively form the basic "RaySense sketch", which can be employed for various data processing tasks.

RaySense does not merely produce narrowly interpreted subsamples of given datasets. Instead, it prioritizes the sampling of salient features of the dataset, such as corners or edges, with a higher probability. Consequently, points near these salient features may be recorded in the sketch tensor multiple times.

From the RaySense sketch, one can further extract snapshots of integral or local (differential) information about the data set. Relevant operations are defined on the rays randomly sampled from a chosen distribution. Since rays are 1D objects, the formal complexity of RaySense does not increase exponentially with the dimensions of the embedding space. We provide theoretical analysis showing that the statistics of a sampled point cloud depends solely on the distribution of the rays, and not on any particular ray set. Additionally, we also demonstrated that by appropriately post-processing the RaySense sketch tensor obtained from a given point cloud, one can compute approximations of line integrals. Thus, and by way of the Fourier Slice Theorem, we argue that RaySense provides spectral information about the sampled point cloud.

We showed that RaySense sketches could be used to register and classify point clouds of different cardinality. For the classification of point clouds in three dimensions, we presented a neural network classifier called "RayNN", which takes the RaySense sketches as input. Nearest-neighbor information can be sensitive to outliers. For finite point sets, we advocated augmentation of the sketch tensor by including multiple nearest neighbors to enhance RaySense's capability to capture persistent features in the data set, thereby improving the robustness. We compared the performance of RayNN to several other prominent models, highlighting its lightweight, flexible, and efficient nature. Importantly, RayNN also differs from conventional models, as it allows for multiple tests with different ray sets on the same dataset.

Appendix A Examples of Ray Distributions

We assume all points are properly calibrated by a common preprocessing step. This could also be learned. In fact, one can use RaySense to train such a preprocessor to register the dataset, for example, using Sect. 4.4 or similar. However, for simplicity, in our experiments, we generally normalize each point set to be in the unit ℓ^2 ball, with a center of mass at the origin, unless otherwise indicated.

We present two ways to generate random rays. There is no *right* way to generate rays, although it is conceivable that one may find optimal ray distributions for specific applications.

Method R1 One simple approach is generating rays of the fixed-length L, whose direction $\boldsymbol{v}$ is uniformly sampled from the unit sphere. We add a shift $\boldsymbol{b}$ sampled uniformly from $[-\frac{1}{2}, \frac{1}{2}]^d$ to avoid a bias for the origin. The n_r sample points are distributed evenly along the ray:

$$\boldsymbol{r}_i = \boldsymbol{b} + L\left(\frac{i}{n_r - 1} - \frac{1}{2}\right)\boldsymbol{v}, \qquad i = 0, \cdots, n_r - 1.$$

The spacing between adjacent points on each ray is denoted by δr, which is $\frac{L}{n_r - 1}$. We use $L = 2$.

Method R2 Another natural way to generate random rays is by random endpoints selection: choose two random points $\boldsymbol{p}, \boldsymbol{q}$ on a sphere and connect them to form a ray. Then we evenly sample n_r points between $\boldsymbol{p}, \boldsymbol{q}$ on the ray. To avoid overly short rays where information would be redundant, we use a minimum ray-length threshold τ to discard rays. Note that the distance between n_r sample points is different on different rays:

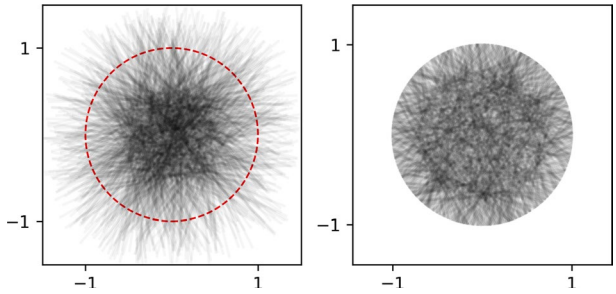


Fig. 13 Density of rays from method R1 (left) and R2 (right). Red circle indicates the ℓ^2 ball

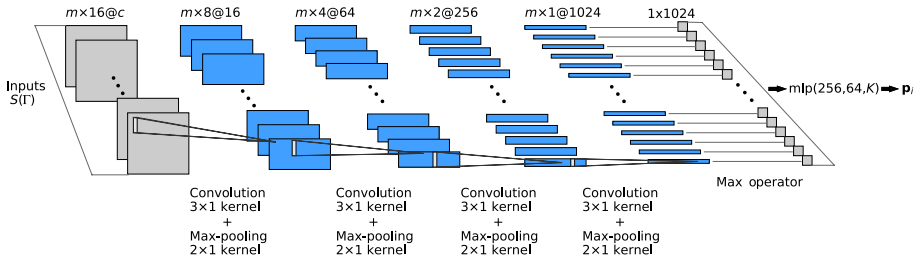


Fig. 14 The RayNN architecture for m rays and n_r samples per ray. The input is c feature matrices from $\mathcal{S}(\Gamma)$ with suitable operations. With $n_r = 16$, each matrix is downsized to an m-vector by 4 layers of 1D convolution and max-pooling. The max operator is then applied to each of the 1 024 m-vectors. The length-1 024 feature vector is fed into a multi-layer perceptron (mlp) which outputs a vector of probabilities, one for each of the K classes in the classification task. Note the number of intermediate layers (blue) can be increased based on n_r and c

$$\boldsymbol{r}_i = \boldsymbol{p} + \frac{i}{n_r - 1}(\boldsymbol{q} - \boldsymbol{p}), \qquad i = 0, \cdots, n_r - 1.$$

The spacing of points on each ray varies, depending on the length of the ray.

Figure 13 shows the density of rays from the ray generation methods. In this paper, we use Method R1; a fixed δr seems to help maintain spatial consistency along the rays, which increases RayNN's classification accuracy in Sect. 4.5.

Appendix B Implementation Details of RayNN

Our implementation uses PyTorch [41].

Architecture RayNN takes the $m \times k \times c$ RaySense sketch tensor $\mathcal{S}(\Gamma)$ as input, and outputs a K-vector of probabilities, where K is the number of object classes.

The first few layers of the network are blocks of 1D convolution followed by max-pooling to encode the sketch into a single vector per ray. Convolution and max-pooling are applied along the ray. After this downsizing, we implemented a max operation across rays. Figure 14 includes some details. The output of the max pooling layer is fed into fully connected layers with output sizes 256, 64, and K to produce the desired vector of probabilities

$\mathbf{p}_i \in \mathbb{R}^K$. Batchnorm [20] along with ReLU [38] is used for every fully-connected and convolution layer.

Note that our network uses convolution along rays to capture local information while the fully connected layers aggregate global information. Between the two, the max operation across rays ensures invariance to the ordering of the rays. It also allows for an arbitrary number of rays to be used during inference. These invariance properties are similar to PointNet's input-order invariance [44].

Data We apply RayNN on the standard ModelNet10 and ModelNet40 benchmarks [62] for 3D object classification. ModelNet40 consists of 12 311 orientation-aligned [50] meshed 3D CAD models, divided into 9 843 training and 2 468 test objects. ModelNet10 contains 3 991 training and 908 test objects. Following the experiment setup in [44], we sample $N = 1\ 024$ points from each of these models and rescale them to be bounded by the unit sphere to form point sets.[1] Our results do not appear to be sensitive to N.

Training During training, we use dropout with ratio 0.5 on the penultimate fully-connected layer. We also augment our training dataset on-the-fly by adding $\mathcal{N}(0, 0.000\ 4)$ noise to the coordinates. For the optimizer, we use Adam [25] with momentum 0.9 and batch size 16. The learning rate starts at 0.002 and is halved every 100 epochs.

Inference Our algorithm uses random rays, so it is natural to consider strategies to reduce the variance in the prediction. We consider one simple approach during inference by making an ensemble of predictions from λ different ray sets. The ensemble prediction is based on the average over the λ different probability vectors $\mathbf{p}_i \in \mathbb{R}^K$, i.e.,

$$\text{Prediction}(\lambda) = \frac{1}{\lambda} \sum_{i=1}^{\lambda} \mathbf{p}_i.$$

The assigned label then corresponds to the entry with the largest probability. We denote the number of rays used during training by m, while the number of rays used for inference is $\hat{m}$. Unless otherwise specified, we use $\lambda = 8$, $m = 32$ rays, and $\hat{m} = m$.

Appendix C Details of the Proof of Theorem 1

This appendix contains the proofs of Lemmas 1, 2, and 3, and Theorem 1.

Proof of Lemma 1 The probability measure of Ω_j is

$$P_{\Omega_j} = \int_{\boldsymbol{x} \in \Omega_j} \rho(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} > 0,$$

which represents the probability of sampling Ω_j when drawing *i.i.d.* random samples from μ. For a fixed set of such hypercubes, any $\boldsymbol{x} \in \text{supp}(\rho)$ will fall in one of the Ω_j's. Then one can define a mapping $h : \text{supp}(\rho) \subset \mathbb{R}^d \to \mathbb{R}$ by

[1] RaySense does not require point clouds for inputs: we could apply RaySense directly to surface meshes, implicit surfaces, or even—given an fast nearest neighbor calculator—the CAD models directly.

$$s = h(\boldsymbol{x}) = j - 1, \quad \text{where } \boldsymbol{x} \in \Omega_j,\ j = 1, 2, \cdots, M.$$

By applying the mapping to the random vector $\boldsymbol{X}$, we obtain a new discrete random variable S with the discrete probability distribution μ_M on $\mathbb{R}$ and the corresponding density ρ_M. The random variable S lives in a discrete space $S \in \{0, 1, \cdots, M-1\}$ and ρ_M is given as a sum of delta spikes as

$$\rho_M(s) = \sum_{j=1}^{M} P_{\Omega_j} \delta_j(s).$$

As a result, sampling from the distribution μ_M is equivalent to sampling the hypercubes according to the distribution μ in $\mathbb{R}^d$, but one cares only about the sample being in a specific hypercube Ω_j, not the precise location of the sample. Let $F_M(s)$ denote the cumulative density function related to the density function $\rho_M(s)$.

Now, given a set of N independent samples of $\boldsymbol{X}$: $\{\boldsymbol{X}_i\}_{i=1}^N \subset \mathbb{R}^d$, we have a corresponding set of N independent sample points of S: $\{s_i\}_{i=1}^N$ such that $\boldsymbol{X}_i \in \Omega_{s_i+1}$. From there, we can regard the histogram of $\{s_i\}_{i=1}^N$ as an empirical density of the true density ρ_M. Denote the empirical density by $\tilde{\rho}_M^N$ which is given by

$$\tilde{\rho}_M^N = \frac{1}{N} \sum_{i=1}^{N} \delta_{s_i}.$$

One can therefore also obtain an empirical cumulative density function $\tilde{F}_M^N(s)$ using the indicator function χ:

$$\tilde{F}_M^N(s) = \frac{1}{N} \sum_{i=1}^{N} \chi_{\{s_i \leqslant s\}}.$$

By Dvoretzky-Kiefer-Wolfowitz inequality [13, 36] we have

$$Prob\left(\sup_{s \in \mathbb{R}} \left| F_M(s) - \tilde{F}_M^N(s) \right| > \varepsilon \right) \leqslant 2\mathrm{e}^{-2N\varepsilon^2} \quad \text{for all } \varepsilon > 0.$$

Therefore, for a desired fixed probability p_0, the above indicates the approximating error given by the empirical $\tilde{F}_M^N(s)$ is at most

$$\sup_{s \in \mathbb{R}} \left| F_M(s) - \tilde{F}_M^N(s) \right| \leqslant \varepsilon_N = \left(-\frac{1}{2N} \ln\left(\frac{1 - p_0}{2} \right) \right)^{\frac{1}{2}}$$

with probability at least p_0. Then note that the true probability measure P_{Ω_j} of Ω_j being sampled by random drawings from μ is equivalent to the true probability of $j-1$ being drawn from μ_M, *i.e.*,

$$P_{\Omega_j} = P_M(j-1) := \rho_M(j-1),$$

therefore, $P_{\Omega_j} = P_M(j-1)$ can be computed from F_M by

$$\begin{aligned} P_{\Omega_j} &= F_M(j-1) - F_M(j-2) \\ &= F_M(j-1) - \tilde{F}_M^N(j-1) + \tilde{F}_M^N(j-1) - \tilde{F}_M^N(j-2) + \tilde{F}_M^N(j-2) - F_M(j-2). \end{aligned}$$

Taking absolute value and using the triangle inequality, with the fixed p_0

$$P_{\Omega_j} \leqslant 2\varepsilon_N + \tilde{P}_M^N(j-1),$$

where $\tilde{P}_M^N(j-1)$ denotes the empirical probability at $j-1$. Applying the same argument to $\tilde{P}_M^N(j-1)$, one has

$$|P_{\Omega_j} - \tilde{P}_M^N(j-1)| \leqslant 2\varepsilon_N \quad \text{for all } j = 1, 2, \cdots, M.$$

For a set of N sample points, $\tilde{P}_M^N(j-1)$ is computed by $\frac{N_j}{N}$, where N_j is the number of times $j-1$ got sampled by $\{s_i\}_{i=1}^N$, or equivalently Ω_j got sampled by $\{\boldsymbol{X}_i\}_{i=1}^N$, which indicates that in practice, with probability at least p_0, the number of sampling points N_j in Ω_j satisfies the following bound:

$$P_{\Omega_j} - 2\varepsilon_N \leqslant \frac{1}{N}N_j \leqslant P_{\Omega_j} + 2\varepsilon_N \implies P_{\Omega_j}N - 2\varepsilon_N N \leqslant N_j \leqslant P_{\Omega_j}N + 2\varepsilon_N N.$$

By taking N large enough such that $P_{\Omega_j}N - 2\varepsilon_N N = 1 \implies N_j \geqslant 1$:

$$\implies N = \frac{\sqrt{\ln\left(\frac{1-p_0}{2}\right)\left(\ln\left(\frac{1-p_0}{2}\right) - 2P_{\Omega_j}\right)} + P_{\Omega_j} - \ln\left(\frac{1-p_0}{2}\right)}{P_{\Omega_j}^2}.$$

The above quantity is clearly a function with respect to the probability measure P_{Ω_j}, and any Ω_i with $P_{\Omega_i} \geqslant P_{\Omega_j}$ would have $N_i \geqslant N_j \geqslant 1$. Using ν to denote such a function and $0 < P \leqslant 1$ as the threshold measure completes the first part of the proof:

$$\implies \nu(P) = \frac{\sqrt{\ln\left(\frac{2}{1-p_0}\right)\left(\ln\left(\frac{2}{1-p_0}\right) + 2P\right)} + P + \ln\left(\frac{2}{1-p_0}\right)}{P^2}.$$

To establish the bounds on the expression, we note

$$\nu(P) > \frac{\sqrt{\ln\left(\frac{2}{1-p_0}\right)\ln\left(\frac{2}{1-p_0}\right)} + P + \ln\left(\frac{2}{1-p_0}\right)}{P^2} = \frac{2\ln\left(\frac{2}{1-p_0}\right) + P}{P^2},$$

$$\nu(P) < \frac{\sqrt{\left(\ln\left(\frac{2}{1-p_0}\right) + 2P\right)^2} + P + \ln\left(\frac{2}{1-p_0}\right)}{P^2} = \frac{2\ln\left(\frac{2}{1-p_0}\right) + 3P}{P^2}.$$

Proof of Lemma 2 Consider a local hypercube centered at $\boldsymbol{y}$, $\Omega_{\boldsymbol{y}} := \{\boldsymbol{x} + \boldsymbol{y} \in \mathbb{R}^d : \|\boldsymbol{x}\|_\infty = \frac{l}{2}\}$ of length l to be determined. We shall just say "cube". The probability of cube $\Omega_{\boldsymbol{y}}$ being sampled is given by $P_{\Omega_{\boldsymbol{y}}} = \int_{\Omega_{\boldsymbol{y}}} \rho(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$. Now for the set of standard basis vector $\{\boldsymbol{e}_i\}_{i=1}^d$, let $\boldsymbol{v}_d$ denote the sum of all the basis: $\boldsymbol{v}_d := \sum_{i=1}^d \boldsymbol{e}_i$. Without loss of generality, the probability of a diagonal cube, defined by $\Omega_{\boldsymbol{y}_d} := \{\boldsymbol{x} + \boldsymbol{y} + \boldsymbol{v}_d \in \mathbb{R}^d : \|\boldsymbol{x}\|_\infty = \frac{l}{2}\}$, being sampled (unconditional to $\Omega_{\boldsymbol{y}}$ being sampled) has the following bound by Lipschitz continuity of ρ:

$$\left|P_{\Omega_{y_d}} - P_{\Omega_y}\right| \leqslant \int_{\Omega_y} |\rho(\boldsymbol{x} + l\boldsymbol{v}_d) - \rho(\boldsymbol{x})| \mathrm{d}\boldsymbol{x} \leqslant L\sqrt{d}\, l|\Omega_y| \implies P_{\Omega_{y_d}} \geqslant P_{\Omega_y} - L\sqrt{d}\, l^{d+1}.$$

Furthermore, P_{Ω_y} has the following lower bound also by Lipschitz continuity of ρ. For any $x \in \Omega_y$, we have

$$|\rho(\boldsymbol{x}) - \rho(\boldsymbol{y})| \leqslant L\sqrt{d}\frac{l}{2} \implies \rho(\boldsymbol{x}) \geqslant \rho(\boldsymbol{y}) - L\frac{\sqrt{d}}{2}l \implies P_{\Omega_y} \geqslant \left(\rho(\boldsymbol{y}) - L\frac{\sqrt{d}}{2}l\right)l^d. \tag{21}$$

Combining with the previous bound for $P_{\Omega_{y_d}}$, we further have

$$P_{\Omega_{y_d}} \geqslant \left(\rho(\boldsymbol{y}) - L\frac{\sqrt{d}}{2}l\right)l^d - L\sqrt{d}l^{d+1} = \rho(\boldsymbol{y})l^d - \frac{3\sqrt{d}}{2}Ll^{d+1}.$$

By setting $\rho(\boldsymbol{y}) > \frac{3\sqrt{d}}{2}Ll$ we can ensure $P_{\Omega_{y_d}} > 0$, but this extreme lower bound is based on on Lipschitz continuity. To obtain a more useful bound, we will show below that by picking $l := l_N$ judiciously, $\rho(\boldsymbol{y}) > 3\sqrt{d}Ll_N > 0$, any surrounding cube has non-zero probability to be sampled. Therefore, with $\rho(\boldsymbol{y}) > 3\sqrt{d}Ll_N$, for any diagonal cube Ω_{y_d}:

$$\rho(\boldsymbol{y})l^d - \frac{1}{2}\rho(\boldsymbol{y})l^d > \frac{3\sqrt{d}}{2}Ll^{d+1} \implies P_{\Omega_{y_d}} > \frac{1}{2}\rho(\boldsymbol{y})l_N^d.$$

Since the diagonal cube is the furthest to $\boldsymbol{y}$ among all the surrounding cubes, we have for every surrounding cube of $\Omega_{\boldsymbol{y}}$, their probability measure is at least $P_{\Omega_{y_d}}$.

According to Lemma 1, for N sampling points, with probability at least p_0, if a region has probability measure $\geqslant P_N$, then there is at least one point sampled in that region, where P_N is the threshold probability depending on N obtained by solving the equation below:

$$N = \frac{\sqrt{\ln\left(\frac{2}{1-p_0}\right)\left(\ln\left(\frac{2}{1-p_0}\right) + 2P_N\right)} + P_N + \ln\left(\frac{2}{1-p_0}\right)}{(P_N)^2}.$$

By the bounds for N in (17) of Lemma 1, we know there is some constant $c \in (1, 3)$ *s.t.*:

$$N = \frac{2\ln\left(\frac{2}{1-p_0}\right) + cP_N}{P_N^2} \implies NP_N^2 - cP_N - 2\ln\left(\frac{2}{1-p_0}\right) = 0.$$

Solving the above quadratic equation and realize that $P_N > 0$, we have

$$P_N = \frac{c + \left(c^2 + 8N\ln\left(\frac{2}{1-p_0}\right)\right)^{\frac{1}{2}}}{2N}.$$

Therefore, for a fixed N, by requiring

$$P_{\Omega_{y_d}} > \frac{\rho(\boldsymbol{y})}{2} l_N^d \geqslant P_N \implies l_N \geqslant \left(\frac{c + \left(c^2 + 8N \ln\left(\frac{2}{1-p_0} \right) \right)^{\frac{1}{2}}}{\rho(\boldsymbol{y}) N} \right)^{\frac{1}{d}},$$

we have with probability p_0 that at every surrounding cube of $\Omega_{\boldsymbol{y}}$ of side length l_N, there is at least one point. This lower bound for l_N ensures the surrounding cube has enough probability measure to be sampled. Since $1 < c < 3$, we can just take l_N to be

$$l_N := \left(\frac{3 + \left(9 + 8N \ln\left(\frac{2}{1-p_0} \right) \right)^{\frac{1}{2}}}{\rho(\boldsymbol{y}) N} \right)^{\frac{1}{d}} > \left(\frac{c + \left(c^2 + 8N \ln\left(\frac{2}{1-p_0} \right) \right)^{\frac{1}{2}}}{\rho(\boldsymbol{y}) N} \right)^{\frac{1}{d}}.$$

From above we see that for a fixed $\rho(\boldsymbol{y})$, l_N decreases as N increases. Therefore, by choosing N large enough, we can always satisfy the prescribed assumption $\rho(\boldsymbol{y}) \geqslant 3\sqrt{d} L l_N$.

Furthermore, when N is so large such that $\rho(\boldsymbol{y}) \geqslant 3\sqrt{d} L l_N$ is always satisfied, we see that l_N is a decreasing function of ρ, meaning that with a higher local density $\rho(\boldsymbol{y})$, the l_N can be taken smaller while the sampling statement still holds, meaning the local region is more compact.

Finally, since there is a point in every surrounding cube of $\Omega_{\boldsymbol{y}}$, the diameter of the Voronoi cell of $\boldsymbol{y}$ has the following upper-bound with the desired probability p_0:

$$\operatorname{diam}(V(\boldsymbol{y})) \leqslant 3l\sqrt{d} = 3\sqrt{d} \left(\frac{3 + \left(9 + 8N \ln\left(\frac{2}{1-p_0} \right) \right)^{\frac{1}{2}}}{\rho(\boldsymbol{y}) N} \right)^{\frac{1}{d}}.$$

Now, for a sample point x_0 in the interior of $\operatorname{supp}(\rho)$, given a cover of cubes as in Lemma 1, x_0 must belong to one of the cubes with center also denoted by $\boldsymbol{y}$ with a slight abuse of notation. Then note that the diameter of $V(\boldsymbol{x}_0)$ also has the same upper bound as shown above. To go from $\rho(\boldsymbol{y})$ to $\rho(\boldsymbol{x}_0)$, by Lipschitz continuity: $\rho(\boldsymbol{y}) \geqslant \rho(\boldsymbol{x}_0) - \frac{L\sqrt{d}}{2} l_N \implies \rho(\boldsymbol{x}_0) \leqslant \rho(\boldsymbol{y}) + \frac{L\sqrt{d}}{2} l_N$. Since we require $\rho(\boldsymbol{y}) \geqslant 3\sqrt{d} L l_N$, we have $\rho(\boldsymbol{x}_0) \leqslant \frac{\rho(\boldsymbol{y})}{6} + \rho(\boldsymbol{y}) = \frac{7}{6}\rho(\boldsymbol{y})$. Therefore,

$$\rho(\boldsymbol{y}) \geqslant \frac{6}{7}\rho(\boldsymbol{x}_0) \implies \operatorname{diam}(V(\boldsymbol{x}_0)) \leqslant 3\sqrt{d} \left(\frac{21 + 7\left(9 + 8N \ln\left(\frac{2}{1-p_0} \right) \right)^{\frac{1}{2}}}{6\rho(\boldsymbol{x}) N} \right)^{\frac{1}{d}}.$$

Proof of Lemma 3 Without loss of generality, we assume that $|\operatorname{supp}(\rho)| = 1$, then $\rho = 1$ everywhere within its support. We partition $\operatorname{supp}(\rho)$ into M *regions* such that each region has probability measure $\frac{1}{M}$. This partition can be constructed in the following way: for most of

the interior of supp(ρ), subdivide into hypercubes Ω_j's of the same size such that $P_{\Omega_j} = \frac{1}{M}$ and Ω_j's are contained completely inside supp(ρ). Then the length of the hypercube, l, is determined by $\frac{l^d}{|\text{supp}(\rho)|} = \frac{1}{M} \implies l = \left(\frac{1}{M^{\frac{1}{d}}}\right)$. For the remaining uncovered regions of supp(ρ), cover with some small cubes of appropriate sizes and combine them together to obtain a region with measure $\frac{1}{M}$.

Then, following a similar idea from Lemma 2, one has a discrete sampling problem with equal probability for each candidate, which resembles the coupon collector problem. The probability $p(N, d, M)$ that each of the M region contains at least one sample point has a well-known lower bound [11]:

$$p(N, d, M) \geqslant 1 - M\mathrm{e}^{-\frac{N}{M}}.$$

With the probability $p(N, d, M)$ given above, for an interior hypercube we again have there is at least one sample in each of its surrounding hypercube, since now there is at least one sample in each of the M region. Then the Voronoi diameter for each point is at most $3l\sqrt{d}$. Fixing a desired probability p_0, we want to determine the number of regions M to get a control on l. We need to have a bound as follows:

$$p \geqslant 1 - M\mathrm{e}^{-\frac{N}{M}} \geqslant p_0 \implies 0 < M\mathrm{e}^{-\frac{N}{M}} \leqslant 1 - p_0. \tag{22}$$

By rearranging, the above equality holds only when

$$\frac{N}{M}\mathrm{e}^{\frac{N}{M}} = \frac{N}{1 - p_0}.$$

The above equation is solvable by using the Lambert W function:

$$M = \frac{N}{W_0\left(\frac{N}{1-p_0}\right)},$$

where W_0 is the principal branch of the Lambert W function. Note that the Lambert W function satisfies

$$W_0(x)\mathrm{e}^{W_0(x)} = x \implies \frac{x}{W_0(x)} = \mathrm{e}^{W_0(x)}.$$

Pluging in the above identity, one has

$$\frac{M}{1 - p_0} = \frac{N}{(1 - p_0)W_0\left(\frac{N}{1-p_0}\right)} \implies M = (1 - p_0)\mathrm{e}^{W_0\left(\frac{N}{1-p_0}\right)}.$$

Also note that the function $M\mathrm{e}^{-\frac{N}{M}}$ is monotonically increasing in M (for $M > 0$), so for the bound in (22) to hold we require

$$M \leqslant (1 - p_0)\mathrm{e}^{W_0\left(\frac{N}{1-p_0}\right)}.$$

By taking the largest possible integer M satisfying the above inequality, we then have

$$l = \left(\frac{1}{M}\right)^{\frac{1}{d}} = \left(\lfloor (1-p_0)\mathrm{e}^{W_0\left(\frac{N}{1-p_0}\right)}\rfloor\right)^{\frac{1}{d}}$$

for every hypercube contained in $\mathrm{supp}(\rho)$. Then this yields a uniform bound for the Voronoi diameter of any point that is in an interior hypercube surrounded by other interior hypercubes:

$$\mathrm{diam}(V) \leqslant 3\sqrt{d}\left(\lfloor (1-p_0)\mathrm{e}^{W_0\left(\frac{N}{1-p_0}\right)}\rfloor\right)^{-\frac{1}{d}}.$$

In terms of the limiting behavior, for large x, the Lambert W function is asymptotic to the following [6, 10]:

$$W_0(x) = \ln x - \ln\ln x + o(1) \implies \mathrm{e}^{W_0(x)} = c(x)\frac{x}{\ln x}$$

with $c(x) \to 1$ as $x \to \infty$. Therefore, for sufficiently large $\frac{N}{(1-p_0)}$, we have

$$\mathrm{diam}(V) \leqslant 3\sqrt{d}\left(\left\lfloor (1-p_0)c\frac{N}{(1-p_0)\ln\frac{N}{1-p_0}}\right\rfloor\right)^{-\frac{1}{d}} = 3\sqrt{d}\left(\left\lfloor \frac{1}{cN}\ln\frac{N}{1-p_0}\right\rfloor\right)^{\frac{1}{d}}.$$

Proof of Theorem 1 Note that when using one ray: $S[\Gamma_1](1,j) = \boldsymbol{x}_{1[j]}$ and $S[\Gamma_2](1,j) = \boldsymbol{x}_{2[j]}$ for $j = 1, 2, \cdots, n_r$. The main idea is to bound the difference between each pair of points using the results introduced in the previous lemmas. Consider a fixed sampling point $\boldsymbol{r}_{1,j} \in \boldsymbol{r}(s)$ whose corresponding closest points are $\boldsymbol{x}_{1[j]}$ and $\boldsymbol{x}_{2[j]}$ in Γ_1 and Γ_2, respectively. We consider two cases: first when $\boldsymbol{r}_{1,j}$ is interior to $\mathrm{supp}(\rho)$, in which case from Lemmas 2 and 3, with probability p_0 we have a bound for the diameter of the Voronoi cell of any interior $\boldsymbol{x}$, denote it by $D(\boldsymbol{x})$ where ρ, p_0, N, and d are assumed to be fixed. Therefore,

$$\|\boldsymbol{x}_{1[j]} - \boldsymbol{r}_{1,j}\|_2 \leqslant D(\boldsymbol{x}_{1[j]}); \quad \|\boldsymbol{x}_{2[j]} - \boldsymbol{r}_{1,j}\|_2 \leqslant D(\boldsymbol{x}_{2[j]}).$$

Then by the triangle inequality: $\|\boldsymbol{x}_{1[j]} - \boldsymbol{x}_{2[j]}\|_2 \leqslant D(\boldsymbol{x}_{1[j]}) + D(\boldsymbol{x}_{2[j]})$, which applies for all sampling points $\boldsymbol{r}_{1,j}$'s in the interior of $\mathrm{supp}(\rho)$, and as $N \to \infty$ we have $D \to 0$ in a rate derived in Lemma 2.

In the case where sampling point $\boldsymbol{r}_{1,j} \in \boldsymbol{r}(s)$ is outside of $\mathrm{supp}(\rho)$, since $\mathrm{supp}(\rho)$ is convex, the closest point to $\boldsymbol{r}_{1,j}$ from $\mathrm{supp}(\rho)$ is always unique, denoted by $\boldsymbol{x}_\rho$. Then, choose R_1 depending on N_1, N_2 such that the probability measure $P_1 = P\big(B_{R_1}(\boldsymbol{x}_\rho) \cap \mathrm{supp}(\rho)\big)$ achieves the threshold introduced in Lemma 1 so that there is at least $\boldsymbol{x}_{\rho,1} \in \Gamma_1$ and $\boldsymbol{x}_{\rho,2} \in \Gamma_2$ that lies in $B_{R_1}(\boldsymbol{x}_\rho) \cap \mathrm{supp}(\rho)$. For sufficiently large N, $\boldsymbol{x}_{1[j]}$ and $\boldsymbol{x}_{2[j]}$ would be points inside $B_{R_1}(\boldsymbol{x}_\rho) \cap \mathrm{supp}(\rho)$ since $\mathrm{supp}(\rho)$ is convex. Then we have

$$\|\boldsymbol{x}_{1[j]} - \boldsymbol{x}_{2[j]}\|_2 \leqslant 2R_1,$$

and we can pick N_1, N_2 large to make R_1 as small as desired. Therefore, we have

$$\|\boldsymbol{x}_{1[j]} - \boldsymbol{x}_{2[j]}\|_2 \to 0 \text{ as } N_1, N_2 \to \infty,$$

in probability for both interior sampling points and outer sampling points $\boldsymbol{r}_{1,j}$, and the convergence starts when N_1, N_2 get sufficiently large. Consequently, for the RaySense matrices $S[\Gamma_1]$ and $S[\Gamma_2]$, we can always find N sufficiently large such that

$$\|S[\Gamma_1] - S[\Gamma_2]\|_F = \sqrt{\sum_{i=1}^{n_r} \|\boldsymbol{x}_{1[j]} - \boldsymbol{x}_{2[j]}\|_2^2} \leqslant \varepsilon$$

for arbitrarily small ε depending on N, n_r, d, and the geometry of supp(ρ).

Remark 3 In case of non-convex supp(ρ) and the sampling point $\boldsymbol{r}_{1,j} \in \boldsymbol{r}(s)$ is outside of supp(ρ), if the ray $\boldsymbol{r}$ is drawn from some distribution $\mathcal{L}$, *with probability one*, $\boldsymbol{r}_{1,j}$ is not equidistant to two or more points on supp(ρ), so the closest point is uniquely determined and we only need to worry about the case that $\boldsymbol{r}_{1,j}$ find the closest point $\boldsymbol{x}_{2[j]}$ from Γ_2 that would be far away from $\boldsymbol{x}_{1[j]}$.

Let $\boldsymbol{x}_\rho$ be the closest point of $\boldsymbol{r}_{1,j}$ from supp(ρ), similarly, choose R_1 depending on N_1, N_2 such that for balls $B_{R_1}(\boldsymbol{x}_{\rho,1})$, the probability measure $P_1 = P\big(B_{R_1}(\boldsymbol{x}_\rho) \cap \text{supp}(\rho)\big)$ achieves the threshold introduced in Lemma 1 so that there is at least $\boldsymbol{x}_{\rho,1} \in \Gamma_1$ and $\boldsymbol{x}_{\rho,2} \in \Gamma_2$ that lies in $B_{R_1}(\boldsymbol{x}_\rho) \cap \text{supp}(\rho)$. Now, consider the case where the closest point $\tilde{\boldsymbol{x}}_\rho$ of $\boldsymbol{r}_{1,j}$ from the partial support supp(ρ) $\setminus B_{R_1}(\boldsymbol{x}_\rho)$ is far from $\boldsymbol{x}_\rho$ due to the non-convex geometry, and denote

$$\delta = \|\boldsymbol{r}_{1,j} - \tilde{\boldsymbol{x}}_\rho\|_2 - \|\boldsymbol{r}_{1,j} - \boldsymbol{x}_\rho\|_2 > 0.$$

We pick N so large that

$$\|\boldsymbol{r}_{1,j} - \boldsymbol{x}_{\rho,1}\| \leqslant \|\boldsymbol{r}_{1,j} - \boldsymbol{x}_\rho\| + R_1 \leqslant \|\boldsymbol{r}_{1,j} - \boldsymbol{x}_{\rho,1}\| + \delta \leqslant \|\boldsymbol{r}_{1,j} - \boldsymbol{x}_{\rho,2}\|,$$

implying we can find $\boldsymbol{x}_{\rho,1}, \boldsymbol{x}_{\rho,2}$ from Γ_1 and Γ_2 closer than $\boldsymbol{x}_{\rho,2}$ from the continuum. Therefore, for sufficiently large N_1 and N_2, we can find both closest points $\boldsymbol{x}_{1[j]}$, $\boldsymbol{x}_{2[j]}$ of $\boldsymbol{r}_{1,j}$ inside $B_{R_1}(\boldsymbol{x}_{\rho,1}) \cap \text{supp}(\rho)$ from Γ_1 and Γ_2, $\Longrightarrow$ $\|\boldsymbol{x}_{1[j]} - \boldsymbol{x}_{2[j]}\|_2 \leqslant 2R_1$. The rest follows identically as in the previous proof.

Appendix D Details of the Proof of Theorem 3

Before deriving the result, we first take a detour to investigate the problem under the setting of the Poisson point process, as a means of generating points in a uniform distribution.

Appendix D.1 Poisson Point Process

A Poisson point process [9] on Ω is a collection of random points such that the number of points $N_{\Omega'}$ in any bounded measurable subsets Ω_j with measure $\mu(\Omega_j)$ is a Poisson random variable with rate $\lambda|\Omega_j|$ such that $N_j \sim \text{Poi}(\lambda|\Omega_j|)$. In other words, we take N, instead of being fixed, to be a random Poisson variable: $N \sim \text{Poi}(\lambda)$, where the rate parameter λ is a constant. Therefore, the underlying Poisson process is homogeneous and it also enjoys the complete independence property, i.e., the number of points in each disjoint and bounded subregion will be completely independent of all the others.

What follows naturally from these properties is that the spatial locations of points generated by the Poisson process is uniformly distributed. As a result, each realization of the homogeneous Poisson process is a uniform sampling of the underlying space with number of points $N \sim \text{Poi}(\lambda)$.

Below we state a series of useful statistical properties and concentration inequality for the Poisson random variable.

- The Poisson random variable $N \sim \text{Poi}(\lambda)$ has mean and variance both λ:

$$\mathbb{E}(N) = Var(N) = \lambda.$$

- The corresponding probability density function is

$$\mathbb{P}(N = k) = \frac{\mathrm{e}^{-\lambda}\lambda^k}{k!}.$$

- A useful concentration inequality [43] (N scales linearly with λ):

$$\mathbb{P}(N \leqslant \lambda - \varepsilon) \leqslant \mathrm{e}^{-\frac{\varepsilon^2}{2(\lambda+\varepsilon)}} \quad \text{or} \quad \mathbb{P}(|N - \lambda| \geqslant \varepsilon) \leqslant 2\mathrm{e}^{-\frac{\varepsilon^2}{2(\lambda+\varepsilon)}}. \tag{23}$$

Furthermore, one can also derive a Markov-type inequality for the event a Poisson random variable $N \sim \text{Poi}(\lambda)$ is larger than some $a > \lambda$ that is independent of λ, different from (23).

Proposition 2 *For Poisson random variable $N \sim \text{Poi}(\lambda)$, it satisfies the following bound for any constant $a > \lambda$:*

$$\mathbb{P}(N \geqslant a) \leqslant \frac{\mathrm{e}^{(a-\lambda)}\lambda^a}{a^a} \iff \mathbb{P}(N < a) \geqslant 1 - \frac{\mathrm{e}^{(a-\lambda)}\lambda^a}{a^a}. \tag{24}$$

Proof By Markov's inequality:

$$\begin{aligned}\mathbb{P}(N \geqslant a) = \mathbb{P}(\mathrm{e}^{tN} \geqslant \mathrm{e}^{ta}) &\leqslant \inf_{t>0} \frac{\mathbb{E}(\mathrm{e}^{tN})}{\mathrm{e}^{ta}} = \inf_{t>0} \frac{\sum_{k=1}^{\infty} \mathrm{e}^{tk} \frac{\lambda^k \mathrm{e}^{-\lambda}}{k!}}{\mathrm{e}^{ta}} \\ &= \inf_{t>0} \frac{\mathrm{e}^{-\lambda} \sum_{k=1}^{\infty} \frac{(\mathrm{e}^t\lambda)^k}{k!}}{\mathrm{e}^{ta}} = \inf_{t>0} \frac{\mathrm{e}^{-\lambda}\mathrm{e}^{\mathrm{e}^t\lambda}}{\mathrm{e}^{ta}} = \inf_{t>0} \frac{\mathrm{e}^{(\mathrm{e}^t-1)\lambda}}{\mathrm{e}^{ta}}.\end{aligned}$$

To get a tighter bound, we want to minimize the R.H.S.. Let $\zeta = \mathrm{e}^t > 1$. Then we minimize the R.H.S. over ζ:

$$\min_{\zeta>1} \frac{\mathrm{e}^{(\zeta-1)\lambda}}{\zeta^a} \iff \min_{\zeta>1} (\zeta - 1)\lambda - a\log(\zeta).$$

A simple derivative test yields the global minimizer $\zeta = \frac{a}{\lambda} > 1$ since we require $a > \lambda$. Thus,

$$\mathbb{P}(N \geqslant a) \leqslant \frac{\mathrm{e}^{(a-\lambda)}\lambda^a}{a^a} \iff \mathbb{P}(N < a) \geqslant 1 - \frac{\mathrm{e}^{(a-\lambda)}\lambda^a}{a^a}. \tag{25}$$

A direct consequence of (23) is that one can identify N with λ with high probability when λ is large, or equivalently the other way around.

Lemma 4 (Identify N with λ) *A point set of cardinality N^* drawn from a uniform distribution, with high probability, can be regarded as a realization of a Poisson point process with rate λ such that*

$$\mathbb{P}\left(\frac{2N^*}{3} \leqslant \lambda \leqslant 2N^*\right) \geqslant 1 - \mathrm{e}^{-\frac{N^*}{6}} - \mathrm{e}^{-\frac{N^*}{18}}.$$

Proof If $N \sim \mathrm{Poi}(\lambda)$, by taking $\varepsilon = \frac{\lambda}{2}$, from (23) we have

$$\mathbb{P}\left(|N - \lambda| < \frac{\lambda}{2}\right) \geqslant 1 - 2\mathrm{e}^{-\frac{\lambda}{12}} \iff \mathbb{P}\left(\frac{\lambda}{2} < N < \frac{3\lambda}{2}\right) \geqslant 1 - 2\mathrm{e}^{-\frac{\lambda}{12}}.$$

Let $\lambda_u = 2N^*$ as a potential upper bound for λ, while $\lambda_l = \frac{2N^*}{3}$ the potential lower bound. Then for $N_u \sim \mathrm{Poi}(\lambda_u)$ and $N_l \sim \mathrm{Poi}(\lambda_l)$:

$$\mathbb{P}(N_u \leqslant N^*) = \mathbb{P}\left(N_u \leqslant \frac{\lambda}{2}\right) \leqslant \mathrm{e}^{-\frac{\lambda_u}{12}},$$
$$\mathbb{P}(N_l \geqslant N^*) = \mathbb{P}\left(N_l \geqslant \frac{3\lambda_l}{2}\right) \leqslant \mathrm{e}^{-\frac{\lambda_l}{12}}.$$

Therefore, if we have some other Poisson processes with rate $\lambda_1 > \lambda_u$, and $\lambda_2 < \lambda_l$ the probabilities of the corresponding Poisson variables $N_1 \sim \mathrm{Poi}(\lambda_1), N_2 \sim \mathrm{Poi}(\lambda_2)$ to achieve at most (or at least) N^* is bounded by

$$\mathbb{P}(N_1 \leqslant N^*) < \mathbb{P}(N_u \leqslant N^*) \leqslant \mathrm{e}^{-\frac{\lambda_u}{12}} = \mathrm{e}^{-\frac{N^*}{6}},$$
$$\mathbb{P}(N_2 \geqslant N^*) < \mathbb{P}(N_l \geqslant N^*) \leqslant \mathrm{e}^{-\frac{\lambda_l}{12}} = \mathrm{e}^{-\frac{N^*}{18}}.$$

Note that both of the events have a probability decaying to 0 as the observation $N^* \to \infty$, therefore we have a confidence interval of left margin $\mathrm{e}^{-\frac{N^*}{6}}$ and right margin $\mathrm{e}^{-\frac{N^*}{18}}$ to conclude that the Poisson parameter λ behind the observation N^* has the bound

$$\frac{2N^*}{3} \leqslant \lambda \leqslant 2N^*.$$

Since the margins shrink to 0 as $N^* \to \infty$, we can identify λ as cN with some constant c around 1 with high probability.

By Lemma 4, for the remaining, we will approach the proof to Theorem 3 from a Poisson process perspective and derive results with the Poisson parameter λ.

Appendix D.2 Main Ideas of the Proof of Theorem 3

Consider a Poisson process with parameter λ in $\mathrm{supp}(\rho)$ and a corresponding point cloud Γ with cardinality $N \sim \mathrm{Poi}(\lambda)$. Based on previous discussion from Sect. 3.1.3, we assume the ray $\boldsymbol{r}(s)$ is given entirely in the interior of $\mathrm{supp}(\rho)$. From Theorem 2, by denoting $1 \leqslant k_i \leqslant N$ such that $\{\boldsymbol{x}_{k_i}\}_{i=1}^M \subset \Gamma$ are points in Γ sensed by the ray and $V_{k_i} := V(\boldsymbol{x}_{k_i})$, equivalently the line integral error is

$$\left|\int_0^1 g\big(\boldsymbol{r}(s)\big)\mathrm{d}s - \int_0^1 g\big(\boldsymbol{x}_{k(s)}\big)\mathrm{d}s\right| \leqslant J\sum_{i=1}^M \int_0^1 \chi\big(\{\boldsymbol{r}(s) \in V_{k_i}\}\big)\|\boldsymbol{r}(s) - \boldsymbol{x}_{k_i}\|\mathrm{d}s. \quad (26)$$

To bound the above quantity, one needs to bound M the number of Voronoi cells a line goes through, the length of $\boldsymbol{r}(s)$ staying inside V_{k_i} and the distance to the corresponding $\boldsymbol{x}_{k_i}$ for each $\boldsymbol{r}(s)$ altogether. Our *key intuition* is stated as follows.

Divide the ray $\boldsymbol{r}(s)$ of length 1 into segments each of length h, and consider a hypercylinder of height h and radius h centering around each segment. If there is at least one point from Γ in each of the hypercylinders, then no point along $\boldsymbol{r}(s)$ will have its nearest neighbor further than distance $H = \sqrt{2}h$ away from $\boldsymbol{r}(s)$. Therefore, we restrict our focus to Ω, a tubular neighborhood of distance H around $\boldsymbol{r}(s)$-that is, a "baguette-like" region with spherical end caps. N_Ω, the number of points of Γ that are inside Ω, will serve as an upper bound for M (the total number of unique nearest neighbors of $\boldsymbol{r}(s)$ in Γ) while the control of the other two quantities (intersecting length and distances to closest points) comes up naturally.

Undoubtedly M depends on the size of Ω, which is controlled by h. The magnitude of h therefore becomes the crucial factor we need to determine. The following lemma motivates the choice of $h = \lambda^{-\frac{1}{d}+\epsilon}$ for some small $1 \gg \epsilon > 0$.

Lemma 5 *Under A1–A7, for a point cloud of cardinality $N \sim \mathrm{Poi}(\lambda)$ generated from a Poisson point process, and a ray $\boldsymbol{r}(s)$ given entirely in $\mathrm{supp}(\rho)$, the number of points N_Ω in the tubular neighborhood of radius $H = \sqrt{2}h$ around $\boldsymbol{r}(s)$ will be bounded when*

$$h = \lambda^{-\frac{1}{d}+\epsilon}$$

for some small $1 \gg \epsilon > 0$, with probability $\to 1$ as $\lambda \to \infty$.

Proof Note that the baguette region Ω has outer radius H, and hypercylinders of radius h are contained inside Ω. For simplicity we prescribe h such that $Q = \frac{1}{h}$ is an integer, then the baguette region Ω consists of Q number of hypercylinders, denoted by $\{\Omega_j\}_{j=1}^{Q}$ and the remaining region, denoted by Ω_r consisting of an annulus of outer radius H, inner radius h, and two half spheres of radius H on each side. Since each region is disjoint, according to Appendix D.1 the Poisson process with rate λ in $\mathrm{supp}(\rho)$ will have Poisson sub-process in each of the regions in a rate related to their Lesbegue measure, and all the sub-processes are independent.

Now, let $\mathbb{P}_Q$ denote the probability of having at least one point in each Ω_j in $\{\Omega_j\}_{j=1}^{Q}$ while the number of points in each Ω_j is also uniformly bounded by some constant N_Q. Since each Ω_j has the same measure, their corresponding Poisson processes have the identical rate $\lambda_q = |\Omega_1|\lambda$. Let N_j denote the Poisson random variable for Ω_j. Then,

$$\mathbb{P}(N_j \geqslant 1) = 1 - \mathbb{P}(N_j = 0) = 1 - \mathrm{e}^{-\lambda_q}.$$

Combined with (24) by requiring $N_Q > \lambda_q$, this implies

$$\mathbb{P}(N_Q > N_j \geqslant 1) = \mathbb{P}(N_j \geqslant 1) - \mathbb{P}(N_j \geqslant N_Q) \geqslant 1 - \mathrm{e}^{-\lambda_q} - \frac{\mathrm{e}^{(N_Q-\lambda_q)}\lambda_q^{N_Q}}{N_Q^{N_Q}},$$

and hence

$$\mathbb{P}_Q \geqslant \left(1 - \mathrm{e}^{-\lambda_q} - \frac{\mathrm{e}^{(N_Q-\lambda_q)}\lambda_q^{N_Q}}{N_Q^{N_Q}}\right)^Q \geqslant 1 - Q\left(\mathrm{e}^{-\lambda_q} + \frac{\mathrm{e}^{(N_Q-\lambda_q)}\lambda_q^{N_Q}}{N_Q^{N_Q}}\right). \tag{27}$$

The measure of the remaining region Ω_r is $|\Omega_r| = \omega_d H^d + \omega_{d-1}(H^{d-1} - h^{d-1})$, where ω_d is the volume of the unit d-sphere. Therefore the Poisson process on Ω_r has rate $\lambda_r = |\Omega_r|\lambda$. Let N_r denote the corresponding Poisson random variable, again by (24) with $N' > \lambda_r$:

$$\mathbb{P}(N_r < N') \geqslant 1 - \frac{\mathrm{e}^{(N'-\lambda_r)}\lambda_r^{N'}}{N'^{N'}}. \tag{28}$$

Since Ω_r and $\bigcup\{\Omega_j\}_{j=1}^Q$ are disjoint, by independence, the combined probability p_{tot} that all these events happen:

(i) the number of points N_j in each hypercylinder Ω_j is at least 1,
(ii) N_j is uniformly bounded above by some constant N_Q,
(iii) the number of points N_r in the remaining regions $\Omega_r = \Omega - \cup_j\{\Omega_j\}$ is also bounded above by some constant N',

would have the lower bound:

$$\begin{aligned} p_{\text{tot}} &\geqslant \left(1 - \frac{\mathrm{e}^{(N'-\lambda_r)}\lambda_r^{N'}}{N'^{N'}}\right)\left(1 - Q\left(\mathrm{e}^{-\lambda_q} + \frac{\mathrm{e}^{(N_Q-\lambda_q)}\lambda_q^{N_Q}}{N_Q^{N_Q}}\right)\right) \\ &\geqslant 1 - \frac{\mathrm{e}^{(N'-\lambda_r)}\lambda_r^{N'}}{N'^{N'}} - Q\left(\mathrm{e}^{-\lambda_q} + \frac{\mathrm{e}^{(N_Q-\lambda_q)}\lambda_q^{N_Q}}{N_Q^{N_Q}}\right). \end{aligned}$$

Then with probability p_{tot}, we have an upper bound for N_Ω, the total number of points in Ω:

$$N_\Omega \leqslant N' + QN_Q. \tag{29}$$

Apparently N_Ω and p_{tot} are inter-dependent: as we restrict the R.H.S. bound in (29) by choosing a smaller N' or N_Q, the bound for p_{tot} will be loosened. From Lemma 4, we set $N' = \alpha\lambda_r$, $N_Q = \beta\lambda_q$ for some $\alpha, \beta > 1$. Therefore, the next step is to determine the parameter set $\{h, \alpha, \beta\}$ to give a more balanced bound to the R.H.S. in (29) while still ensuring the probability of undesired events will have exponential decay.

For that purpose we need some *optimization*. We know

$$\lambda_r = |\Omega_r|\lambda = (\omega_d H^d + \omega_{d-1}(H^{d-1} - h^{d-1}))\lambda = \left(\omega_d 2^{\frac{d}{2}} h^d + \omega_{d-1}\left(2^{\frac{d-1}{2}} - 1\right)h^{d-1}\right)\lambda;$$

$$\lambda_q = |\Omega_1|\lambda = (\omega_{d-1}h^{d-1}h)\lambda = \omega_{d-1}h^d\lambda. \tag{30}$$

We need to investigate how h should scale with λ, so we assume $h \sim \lambda^{-p}$ for some constant p to be determined. The following optimization procedure provides some motivations for choosing p. On the one hand, for the constraints we need to ensure that the probability of each of the three events above not occurring decays to 0 as $\lambda \to \infty$

$$\begin{aligned}
\frac{\mathrm{e}^{(N'-\lambda_r)}\lambda_r^{N'}}{N'^{N'}} \to 0 &\iff (N'-\lambda_r)+N'\log(\lambda_r)-N'\log(N') \to -\infty,\\
Q\mathrm{e}^{-\lambda_q} \to 0 &\iff \log(Q)-\lambda_q \to -\infty,\\
Q\frac{\mathrm{e}^{(N_Q-\lambda_q)}\lambda_q^{N_Q}}{N_Q^{N_Q}} \to 0 &\iff \log(Q)+(N_Q-\lambda_q)+N_Q\log(\lambda_q)-N_Q\log(N_Q) \to -\infty,
\end{aligned}$$

and representing all the quantities in terms of λ, p, α, β and simplifying

$$\begin{aligned}
(\alpha-1)\lambda_r-\alpha\lambda_r\log(\alpha)\to-\infty \quad &\Longrightarrow\ \alpha\big(1-\log(\alpha)\big)<1 \quad \Longrightarrow\ \alpha>1,\\
p\log(\lambda)-\omega_{d-1}\lambda^{-pd+1}\to-\infty \quad &\Longrightarrow\ \beta\big(1-\log(\beta)\big)<1 \quad \Longrightarrow\ \beta>1,\\
\log(Q)+(\beta-1)\lambda_q-\beta\lambda_q\log(\beta)\to-\infty \quad &\Longrightarrow\ -pd+1>0 \quad \Longrightarrow\ p<\frac{1}{d}.
\end{aligned}$$

On the other hand, for the objective, note that

$$\begin{aligned}
N'+\frac{N_Q}{h} &\leqslant 2\max\left(N',\ \frac{N_Q}{h}\right)\\
&= 2\max\left(\alpha\left(\left(\omega_d 2^{\frac{d}{2}}h^d+\omega_{d-1}\left(2^{\frac{d-1}{2}}-1\right)h^{d-1}\right)\lambda\right),\ \frac{\beta}{h}\omega_{d-1}h^d\lambda\right)\\
&\leqslant 3\max\left(\alpha\omega_d 2^{\frac{d}{2}}h^d\lambda,\ \alpha\omega_{d-1}\left(2^{\frac{d-1}{2}}-1\right)h^{d-1}\lambda,\ \frac{\beta}{h}\omega_{d-1}h^d\lambda\right),
\end{aligned}$$

since α, β are just some constants >1, fixing α and β so that $h=\lambda^{-p}$ and we want to minimize h to obtain an upper bound for the total number of points in Ω:

$$\begin{aligned}
&\arg\min_h \max\left(\alpha\omega_d 2^{\frac{d}{2}}h^d\lambda,\ \alpha\omega_{d-1}\left(2^{\frac{d-1}{2}}-1\right)h^{d-1}\lambda,\ \frac{\beta}{h}\omega_{d-1}h^d\lambda\right)\\
\iff\ &\arg\min_p \max\left(c_2+(1-pd+p)\log(\lambda),\ c_3+(1-pd+p)\log(\lambda)\right)\\
\iff\ &\arg\min_p \max\left((1-pd+p)\log(\lambda)\right).
\end{aligned}$$

Combined with bounds derived from the constraints, to minimize $(1-(d-1)p)$, we need to maximize p, therefore we take $p=\frac{1}{d}-\varepsilon$ for an infinitesimal $\varepsilon>0$.

Appendix D.3 Proof of Theorem 3

Proof of Theorem 3 Consider a Poisson process with rate λ on the $\mathrm{supp}(\rho)$. As in Appendix D.2, let $Q=\frac{1}{h}$ be an integer for simplicity (or take ceiling if desired), and consider Q hypercylinders of radius h centered along $\boldsymbol{r}(s)$. Again as in Appendix D.2, let Ω be the tubular neighborhood of distance $H=\sqrt{2}h$ around $\boldsymbol{r}(s)$. Motivated by Appendix D.2, we set

$$h=\lambda^{-\frac{1}{d}+\varepsilon}$$

for some small constants $1 \gg \varepsilon > 0$ to be determined.

Divide the tubular neighborhood Ω into two parts, one consists of the set of hypercylinders $\bigcup_{j=1}^{Q} \Omega_j$ around $\boldsymbol{r}(s)$, the other is the remainder Ω_r. From the setting of Lemma 5, let $N' = \alpha\lambda_r$ be the number of points in Ω_r while $N_Q = \beta\lambda_q$ is for Ω_j, and we set $\alpha = \beta = \mathrm{e} > 1$ (also satisfying the constraints in Lemma 5) to simplify the calculations so that we have $N_Q = \mathrm{e}\lambda_q, N_r = \mathrm{e}\lambda_r$, and equation (27) becomes

$$\mathbb{P}(\mathrm{e}\lambda_q > N_j \geqslant 1) = \mathbb{P}(N_j \geqslant 1) - \mathbb{P}(N_j \geqslant \mathrm{e}\lambda_q) \geqslant 1 - 2\mathrm{e}^{-\lambda_q},$$

$$\Longrightarrow \mathbb{P}_Q \geqslant \left(1 - 2\mathrm{e}^{-\lambda_q}\right)^Q \geqslant 1 - 2Q\mathrm{e}^{-\lambda_q}. \tag{31}$$

So the total number in $\bigcup_{j=1}^{Q} \Omega_j$ is bounded by $Q\mathrm{e}\lambda_q$ while there is still at least one point in every Ω_j with the above probability. On the other hand for (28),

$$\mathbb{P}(N_r < \mathrm{e}\lambda_r) \geqslant 1 - \mathrm{e}^{-\lambda_r}.$$

Again by the same independence argument, the total probability that all the events happen has the following lower bound:

$$p_{\mathrm{tot}} \geqslant (1 - \mathrm{e}^{-\lambda_r})(1 - 2Q\mathrm{e}^{-\lambda_q}) \geqslant 1 - 2Q\mathrm{e}^{-\lambda_q} - \mathrm{e}^{-\lambda_r}.$$

And the total number of points inside Ω is bounded by

$$N_\Omega \leqslant \mathrm{e}\lambda_r + Qe\lambda_q = \mathrm{e}(\lambda_r + Q\lambda_q) = \mathrm{e}\lambda_\Omega.$$

Finally, when there is at least one point in each of Ω_j, the maximum distance from any point on $\boldsymbol{r}(s)$ to its nearest neighbor is given by $H = \sqrt{2}h$ as we have argued. Furthermore, under this setting, for any of the potential nearest neighbors, the maximum length that $\boldsymbol{r}(s)$ intersect its Voronoi cell has an upper bound of $3h$. Therefore, the line integral error (26) is bounded by

$$\left| \int_0^1 g\big(\boldsymbol{r}(s)\big)\mathrm{d}s - \int_0^1 g\big(\boldsymbol{x}_{k(s)}\big)\mathrm{d}s \right| \leqslant \frac{J}{2} \sum_{i=1}^{M} H \times 3h \leqslant \frac{J}{2} N_\Omega \times H \times 3h = 3\sqrt{2}h^2\mathrm{e}\lambda_\Omega$$

$$\leqslant \frac{3\mathrm{e}\sqrt{2}J}{2} h^2 \left(\omega_d 2^{\frac{d}{2}} h^d + \omega_{d-1} 2^{\frac{d-1}{2}} h^{d-1} \right) \lambda \leqslant c(d, J)(h^{d+2} + h^{d+1})\lambda \leqslant c(d, J)\lambda^{-\frac{1}{d}+\varepsilon(d+1)}.$$

Finally, for the total probability p_{tot}:

$$p_{\mathrm{tot}} \geqslant 1 - 2Q\mathrm{e}^{-\lambda_q} - \mathrm{e}^{-\lambda_r} = 1 - \frac{2}{h}\mathrm{e}^{-\omega_{d-1}h^d\lambda} - \mathrm{e}^{-|\Omega_r|\lambda},$$

and recall from (30): $\lambda_q = |\Omega_1|\lambda = \omega_{d-1}h^{d-1}h\lambda = \omega_{d-1}h^d\lambda = \omega_{d-1}\lambda^{\varepsilon d}$. Then

$$2Q\mathrm{e}^{-\lambda_q} = 2(\lambda)^{\frac{1}{d}-\varepsilon}\mathrm{e}^{-\omega_{d-1}\lambda^{\varepsilon d}} \to 0 \ \text{ as } \lambda \to \infty.$$

The above convergence can be shown by taking the natural log:

$$\ln\left(2(\lambda)^{\frac{1}{d}-\varepsilon}\mathrm{e}^{-\omega_{d-1}\lambda^{\varepsilon}}\right) = \ln 2 + \left(\frac{1}{d} - \varepsilon\right)\ln\lambda - \omega_{d-1}\lambda^{\varepsilon} \to -\infty \ \text{ as } \lambda \to \infty,$$

since $\ln \lambda$ grows slower than λ^{ϵ} for any $\epsilon > 0$. As for the last term $e^{-|\Omega_r|\lambda}$:

$$e^{-|\Omega_r|\lambda} \leqslant e^{-\omega_{d-1}(2^{\frac{d-1}{2}}-1)h^{d-1}\lambda} \leqslant e^{-c\lambda^{\frac{1}{d}+\epsilon(d-1)}} \leqslant e^{-c(d)\lambda^{\frac{1}{d}}} \to 0 \text{ as } \lambda \to \infty.$$

Thus, the probability $p_{\text{tot}} \to 1$ as $\lambda \to \infty$, and we have our line integral error $\leqslant c(d, J)\lambda^{-\frac{1}{d}+\epsilon(d+1)} \to 0$ as long as $\epsilon < \frac{1}{(d+1)^2}$. To obtain the convergence in terms of the actual number of points N in the point cloud, we invoke Lemma 4 and set $N = c\lambda$ to conclude the proof.

Acknowledgements Part of this research was performed while Macdonald and Tsai were visiting the Institute for Pure and Applied Mathematics (IPAM), which is supported by the National Science Foundation (Grant No. DMS-1440415). This work was partially supported by a grant from the Simons Foundation, NSF Grants DMS-1720171 and DMS-2110895, and a Discovery Grant from Natural Sciences and Engineering Research Council of Canada. The authors thank the Texas Advanced Computing Center (TACC) and UBC Math Dept Cloud Computing for providing computing resources.

Data Availability All the datasets used in this paper are well-known public datasets, and they are available through a simple search.

Compliance with Ethical Standards

Conflict of Interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Atzmon, M., Maron, H., Lipman, Y.: Point convolutional neural networks by extension operators. arXiv:1803.10091 (2018)
2. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM 18(9), 509–517 (1975)
3. Besl, P.J., McKay, N.D.: Method for registration of 3-D shapes. In: Sensor Fusion IV: Control Paradigms and Data Structures, vol. 1611 (1992)
4. Caflisch, R.E.: Monte Carlo and quasi-Monte Carlo methods. Acta Numer. **7**, 1–49 (1998)
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J.X., Yi, L., Yu, F.: ShapNet: an information-rich 3D model repository. arXiv:1512.03012 (2015)
6. Corless, R.M., Gonnet, G.H., Hare, D.E., Jeffrey, D.J., Knuth, D.E.: On the Lambert*W* function. Adv. Comput. Math. **5**(1), 329–359 (1996)
7. Curless, B., Levoy, M.: A volumetric method for building complex models from range images computer graphics. In: SIGGRAPH 1996 Proceedings (1996)
8. Cutler, A., Breiman, L.: Archetypal analysis. Technometrics **36**(4), 338–347 (1994)
9. Daley, D.J., Vere-Jones, D.: An Introduction to the Theory of Point Processes. Volume I: Elementary Theory and Methods. Springer, New York (2003)
10. De Bruijn, N.G.: Asymptotic Methods in Analysis, vol. 4. Courier Corporation, USA (1981)
11. Doerr, B.: Probabilistic tools for the analysis of randomized optimization heuristics. In: Theory of Evolutionary Computation, pp. 1–87. Springer, Cham, Switzerland (2020)
12. Draug, C., Gimpel, H., Kalma, A.: The Octave Image package (version 2.14.0) (2022). https://gnu-octave.github.io/packages/image
13. Dvoretzky, A., Kiefer, J., Wolfowitz, J.: Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. Ann. Math. Stat. **27**(3), 642–669 (1956)
14. Eldar, Y., Lindenbaum, M., Porat, M., Zeevi, Y.Y.: The farthest point strategy for progressive image sampling. IEEE Trans. Image Process. **6**(9), 1305–1315 (1997)
15. Fang, Y., Xie, J., Dai, G., Wang, M., Zhu, F., Xu, T., Wong, E.: 3D deep shape descriptor. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2319–2328 (2015)

16. Graham, R., Oberman, A.M.: Approximate convex hulls: sketching the convex hull using curvature. arXiv:1703.01350 (2017)
17. Hadwiger, H.: Vorlesungen Über Inhalt, Oberfläche und Isoperimetrie, vol. 93. Springer, Berlin (1957)
18. Helgason, S., Helgason, S.: The Radon Transform, vol. 2. Springer, New York (1980)
19. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 604–613 (1998)
20. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167 (2015)
21. Jiménez, P., Thomas, F., Torras, C.: 3D collision detection: a survey. Comput. Gr. **25**(2), 269–285 (2001)
22. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2019)
23. Jones, P.W., Osipov, A., Rokhlin, V.: A randomized approximate nearest neighbors algorithm. Applied and Computational Harmonic Analysis **34**(3), 415–444 (2013)
24. Kazmi, I.K., You, L., Zhang, J.J.: A survey of 2D and 3D shape descriptors. In: 2013 10th International Conference Computer Graphics, Imaging and Visualization, pp. 1–10. IEEE (2013)
25. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
26. Klain, D.A., Rota, G.-C.: Introduction to Geometric Probability. Cambridge University Press, Cambridge (1997)
27. Klokov, R., Lempitsky, V.: Escape from cells: deep KD-networks for the recognition of 3D point cloud models. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 863–872 (2017)
28. Krig, S.: Interest point detector and feature descriptor survey. In: Computer Vision Metrics, pp. 187–246. Springer, Cham (2016)
29. LeCun, Y.: The MNIST database of handwritten digits (1998). http://yann.lecun.com/exdb/mnist/
30. Li, J.X., Chen, B.M., Lee, H.: SO-Net: self-organizing network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9397–9406 (2018)
31. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: convolution on X-transformed points. In: Advances in Neural Information Processing Systems, pp. 820–830 (2018)
32. Lin, M., Gottschalk, S.: Collision detection between geometric models: a survey. Proc. IMA Conf. Math. Surf. **1**, 602–608 (1998)
33. Macdonald, C.B., Merriman, B., Ruuth, S.J.: Simple computation of reaction-diffusion processes on point clouds. Proc. Natl. Acad. Sci. **110**(23), 9209–9214 (2013)
34. Macdonald, C.B., Miller, M., Vong, A., et al.: The Octave Symbolic package (version 3.0.1) (2022). https://gnu-octave.github.io/packages/symbolic
35. Mahalanobis, P.C.: On the generalised distance in statistics. Proc. Natl. Inst. Sci. India **2**, 49–55 (1936)
36. Massart, P.: The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. Ann. Prob. **1990**, 1269–1283 (1990)
37. Meurer, A., Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger, B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roučka, Š., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., Scopatz, A.: SymPy: symbolic computing in Python. PeerJ Comput. Sci. **3**, e103 (2017). https://doi.org/10.7717/peerj-cs.103
38. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp. 807–814 (2010)
39. Natterer, F.: The Mathematics of Computerized Tomography. Society for Industrial and Applied Mathematics, USA (2001)
40. Osting, B., Wang, D., Xu, Y., Zosso, D.: Consistency of archetypal analysis. SIAM J. Math. Data Sci. **3**(1), 1–30 (2021).
41. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NIPS Autodiff Workshop (2017)
42. Peyré, G., Cuturi, M., et al.: Computational optimal transport: with applications to data science. Found. Trends® Mach. Learn. **11**(5–6), 355–607 (2019)
43. Pollard, D.: Convergence of Stochastic Processes. Springer, New York (1984)
44. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: deep learning on point sets for 3D classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)

45. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems, pp. 5099–5108 (2017)
46. Radon, J.: über die bestimmung von funktionen durch ihre integralwerte längs gewisser mannigfaltigkeiten. Class. Pap. Mod. Diagn. Radiol. **5**(21), 124 (2005)
47. Rostami, R., Bashiri, F.S., Rostami, B., Yu, Z.: A survey on data-driven 3D shape descriptors. Comput. Graph. Forum **38**(1), 356–393 (2019)
48. Ruuth, S.J., Merriman, B.: A simple embedding method for solving partial differential equations on surfaces. J. Comput. Phys. **227**(3), 1943–1961 (2008)
49. Santaló, L.A.: Integral Geometry and Geometric Probability. Cambridge University Press, New York (2004)
50. Sedaghat, N., Zolfaghari, M., Amiri, E., Brox, T.: Orientation-boosted voxel nets for 3D object recognition. arXiv:1604.03351 (2016)
51. Settles, B.: Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences (2009)
52. Shen, Y., Feng, C., Yang, Y., Tian, D.: Mining point cloud local structures by kernel correlation and graph pooling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
53. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3693–3702 (2017)
54. Sinha, A., Bai, J., Ramani, K.: Deep learning 3D shape surfaces using geometry images. In: European Conference on Computer Vision. Springer, Berlin (2016)
55. Solmon, D.C.: The X-ray transform. J. Math. Anal. Appl. **56**(1), 61–83 (1976)
56. The mpmath development team: mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 1.2.1). (2021). https://mpmath.org/
57. Trefethen, L.N., Weideman, J.A.C.: The exponentially convergent trapezoidal rule. SIAM Rev. **56**(3), 385–458 (2014)
58. Tsai, Y.-H.R.: Rapid and accurate computation of the distance function using grids. J. Comput. Phys. **178**(1), 175–195 (2002)
59. Villani, C.: Optimal Transport: Old and New, vol. 338. Springer, Berlin (2009)
60. Wang, P.-S., Liu, Y., Guo, Y.-X., Sun, C.-Y., Tong, X.: O-CNN: Octree-based convolutional neural networks for 3D shape analysis. ACM Trans. Gr. (TOG) **36**(4), 72 (2017)
61. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Trans. Gr. (TOG) **38**(5), 146 (2019)
62. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D shapenets: a deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1912–1920 (2015)
63. Xia, F., et al.: PointNet.pytorch Git repository. https://github.com/fxia22/pointnet.pytorch
64. Xie, J., Dai, G., Zhu, F., Wong, E.K., Fang, Y.: Deepshape: deep-learned shape descriptor for 3D shape retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **39**(7), 1335–1345 (2016)
65. Zhou, Y., Tuzel, O.: VoxelNet: end-to-end learning for point cloud based 3D object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4490–4499 (2018)