Leaps and Bounds: An Improved Point Cloud Winding Number Formulation for Fast Normal Estimation and Surface Reconstruction

Supplementary Material

A. Derivations

A.1. Deriving the bounded 2D winding number

The 2D winding number from can be rearranged as

$$w_U(q) = \sum_{i=1}^{N} a_i \frac{(p_i - q)^T n_i}{2\pi \|p_i - q\|_2^2} \tag{28}$$

$$= \sum_{i=1}^{N} \frac{a_i^\perp}{2\pi \|p_i - q\|_2} \tag{29}$$

Considering $a_i^\perp$ as a line segment on the tangent line at p_i and symmetric around p_i, this line segment subtends a signed angle θ_i at q related by

$$\tan\left(\frac{\theta}{2}\right) = \frac{\frac{a_i^\perp}{2}}{\|p_i - q\|_2} \tag{30}$$

$$\therefore \theta = 2\arctan\left(\frac{a_i^\perp}{2\|p_i - q\|_2}\right) \tag{31}$$

so the projected arc length is

$$\breve{a_i} = \|p_i - q\|_2 \theta \tag{32}$$

$$= 2\|p_i - q\|_2 \arctan\left(\frac{a_i^\perp}{2\|p_i - q\|_2}\right). \tag{33}$$

Thus our bounded winding number becomes

$$w_B(q) = \sum_i 2\|p_i - q\|_2 \arctan\left(\frac{a_i^\perp}{2\|p_i - q\|_2}\right) \frac{1}{2\pi \|p_i - q\|_2} \tag{34}$$

$$= \sum_i \frac{1}{\pi} \arctan\left(a_i \frac{(p_i - q)^T n_i}{2\|p_i - q\|_2^2}\right). \tag{35}$$

A.2. Winding number gradients

$G(\mu) : \mathbb{R}^{3N} \to \mathbb{R}^{3N}$ is given by

$$G(\mu)_{3i:3(i+1)} = \sum_j \frac{\partial}{\partial p_i} \left(\frac{(p_j - p_i)^T \mu_j}{4\pi \|p_j - p_i\|_2^3}\right) \tag{36}$$

$$= \sum_j \frac{1}{4\pi} \left(\frac{3(p_j - p_i)^T \mu_j}{\|p_j - p_i\|_2^5}(p_j - p_i) - \frac{1}{\|p_j - p_i\|_2^3}\mu_j\right) \tag{37}$$

which can be rearranged into a Hessian vector product as

$$G(\mu)_{3i:3(i+1)} = \sum_j \left(\frac{3}{4\pi \|p_j - p_i\|_2^5}(p_j - p_i)(p_j - p_i)^T - \frac{1}{4\pi \|p_j - p_i\|_2^3} I_3\right) \mu_j. \tag{38}$$

The gradient w.r.t. μ_i is given by

$$\nabla_{\mu_j} w_U(p_i; p_j, \mu_j) = \nabla_{\mu_j} \left(\frac{(p_j - p_i)^T \mu_j}{4\pi \|p_j - p_i\|_2^3} \right) \tag{39}$$

$$= \frac{(p_j - p_i)}{4\pi \|p_j - p_i\|_2^3}, \tag{40}$$

A.3. Bounded winding number gradients

First note that

$$\frac{d}{dx} c(x) = \frac{d}{dx} \left(\frac{1}{2} \mathbf{sign}(x) \left(1 - \frac{1}{\sqrt{|4x| + 1}} \right) \right) \tag{41}$$

$$= \left(\frac{d}{dx} \mathbf{sign}(x) \right) \frac{1}{2} \left(1 - \frac{1}{\sqrt{|4x| + 1}} \right) + \frac{1}{2} \mathbf{sign}(x) \frac{d}{dx} \left(1 - \frac{1}{\sqrt{|4x| + 1}} \right) \tag{42}$$

$$= 2\delta(x) \frac{1}{2} \left(1 - \frac{1}{\sqrt{|4x| + 1}} \right) + \frac{1}{2} \mathbf{sign}(x) \frac{2x}{|x|(1 + |4x|)^{\frac{3}{2}}} \tag{43}$$

$$= 0 + \frac{x \mathbf{sign}(x)}{|x|(1 + |4x|)^{\frac{3}{2}}} \tag{44}$$

$$= \frac{1}{(1 + |4x|)^{\frac{3}{2}}}. \tag{45}$$

$G_B(\mu) : \mathbb{R}^{3N} \to \mathbb{R}^{3N}$ is given by

$$G_B(\mu)_{3i:3(i+1)} = \sum_j \frac{\partial}{\partial p_i} c\left(w_U(p_i, p_j, \mu_j)\right) \tag{46}$$

$$= \sum_j \frac{1}{(1 + |4 w_U(p_i, p_j, \mu_j)|)^{\frac{3}{2}}} \frac{\partial}{\partial p_i} (w_U(p_i, p_j, \mu_j)) \tag{47}$$

$$= \sum_j \frac{1}{(1 + |4 w_U(p_i, p_j, \mu_j)|)^{\frac{3}{2}}} \frac{1}{4\pi} \left(\frac{3(p_j - p_i)^T \mu_i}{\|p_j - p_i\|_2^5} (p_j - p_i) - \frac{1}{\|p_j - p_i\|_2^3} \mu_j \right) \tag{48}$$

$$= \sum_j \frac{1}{4\pi} \frac{1}{(1 + |4 w_U(p_i, p_j, \mu_j)|)^{\frac{3}{2}}} \left(\frac{3(p_j - p_i)^T \mu_i}{\|p_j - p_i\|_2^5} (p_j - p_i) - \frac{1}{\|p_j - p_i\|_2^3} \mu_j \right) \tag{49}$$

which can be rearranged into a Hessian vector product as

$$G_B(\mu)_{3i:3(i+1)} = \sum_j \frac{1}{(1 + |4 w_U(p_i, p_j, \mu_j)|)^{\frac{3}{2}}} \left(\frac{3}{4\pi \|p_j - p_i\|_2^5} (p_j - p_i)(p_j - p_i)^T - \frac{1}{4\pi \|p_j - p_i\|_2^3} I_3 \right) \mu_j. \tag{50}$$

The gradient w.r.t. μ_i is given by

$$\nabla_{\mu_j} w_B(p_i; p_j, \mu_j) = \nabla_{\mu_j} c\left(w_U(p_i, p_j, \mu_j)\right) x_i \tag{51}$$

$$= \frac{1}{(1 + |4 w_U(p_i, p_j, \mu_j)|)^{\frac{3}{2}}} \nabla_{\mu_j} \left(w_U(p_i, p_j, \mu_j)\right) \tag{52}$$

$$= \frac{1}{(1 + |4 w_U(p_i, p_j, \mu_j)|)^{\frac{3}{2}}} \nabla_{\mu_j} \left(\frac{(p_j - p_i)^T \mu_j}{4\pi \|p_j - p_i\|_2^3} \right) \tag{53}$$

$$= \frac{1}{(1 + |4 w_U(p_i, p_j, \mu_j)|)^{\frac{3}{2}}} \frac{(p_j - p_i)}{4\pi \|p_j - p_i\|_2^3}. \tag{54}$$

B. Experimental Details

GCNO dataset. Following GCNO [54], we sample 3k points from the shapes in their subset, and report on percentage of good points (PGP), angle RMSE and chamfer distance. Since we do not have access to their 3k sampling, we run all previous methods on our sampling using their code, rather than use results reported in GCNO. PGP and angle RSME are metrics on the estimated normals for each input point: PGP is defined as the percentage of input points whose estimated normal is within 90° of the ground truth normal, and angle RMSE is the root mean squared error between the true and estimated normals in degrees. For chamfer distance, a mesh is created from the estimated normals using SPSR [28], after which both the ground truth and reconstructed mesh is sampled random with 100k samples each. The chamfer distance is the average of the two one sided chamfer distances, each computing the average distance to the closest point in the other mesh for all points in the original mesh.

We compare against four normal estimation methods, iPSR [20], PGR [34], GCNO [54] and WNNC [35]. Note that PGR, GCNO and WNNC also use winding numbers.

ShapeNet. We use the subset of ShapeNet from Koneputugodage et al. [31], which contains 20 shapes that are determined to be the most complex to reconstruct from the ShapeNet [12] subset provided by [52]. They determine this using their SION metric, which determines how different the surface of the shape is from an enclosing sphere in terms of reachability. The dataset provides 100k uniformly sampled points for each shape as the input for surface reconstruction, normals for each of these points, and also provides 100k points uniform samples from the domain with ground truth labels for whether each domain sample is within the shape or not. Usually chamfer distance and interior IoU is reported, however we replace chamfer distance with angular RSME as it is more discriminative now that the benchmark is saturating. Note that IoU is computed using the labeled domain samples given in the dataset. We again use SPSR to reconstruct a surface from our estimated normals.

We compare to previous unoriented surface reconstruction methods on this dataset, SAP [49] and PG-SDF [31], and two normal estimation methods, iPSR [20] and WNNC [35]. Note that PGR and GCNO are unable to run on input points clouds of this size, PGR's linear system does not fit within GPU memory (we used a 24GB RTX3090) and GCNO does not within within 24hrs (depending on the shape it even gets stuck on building the Voronoi diagram).

C. Further Results

We show results on various splits from the SCUT surface reconstruction benchmark [25], including 20 real scans in Tab. 4, and 30 synthetic shapes with various artifacts to them in Tabs. 5 to 8. These are no artifacts (perfect), non-uniform sampling, noise and missing data. For perfect and non-uniform sampling we categorize the 30 shapes by the three levels of shape complexity that the dataset identifies. For noise and missing data, the dataset gives three levels of the artifact, which is applied to all 30 shapes. We use their provided metrics: Normal Consistency Score (NCS), Chamfer Distance (CD) and F-score (Fs).

We compare the original WNNC algorithm, WNNC (Ω), with our bounded formulation of WNNC, WNNC (SD, A, G_B, Ω). We use the smallest smoothing width schedule, Ω_0, and also use Ω_1 when the artifacts are more pronounced.

On real scans our bounded formulation is consistently better for both noise levels on all metrics. For perfect data, both methods achieve the same results with one minor deviation. For non-uniform data, the results are similar except for complex shapes where our bounded formulation is better. For noisy shapes our bounded formulation is comparative for lower noise but better for higher amount of noise, especially when using the lower smoothing widths. Finally, for missing data the results are comparative at all levels and smoothing widths, with no method doing clearly better in some instance.

	NCS ↑		CD ↓		Fs ↑	
Method	Mean	Std	Mean	Std	Mean	Std
WNNC (Ω_0)	0.9275	0.0778	36.4386	31.3410	0.8540	0.1704
WNNC (SD, A, G_B, Ω_0)	**0.9282**	0.0771	**35.6997**	30.6559	**0.8554**	0.1695
WNNC (Ω_1)	0.9425	0.0524	35.2982	30.0446	0.8616	0.1635
WNNC (SD, A, G_B, Ω_1)	**0.9427**	0.0520	**35.2618**	30.0302	**0.8620**	0.1630

Table 4. Results on real-scanned data (20 shapes) from the SCUT surface reconstruction benchmark [25].

	Simple			Ordinary			Complex		
Method	NCS	CD	Fs	NCS	CD	Fs	NCS	CD	Fs
WNNC (Ω_0)	0.988	0.239	0.971	0.970	0.145	0.999	0.949	**0.177**	0.975
WNNC (SD, A, G_B, Ω_0)	0.988	0.239	0.971	0.970	0.145	0.999	0.949	0.182	0.975

Table 5. Results for perfect data, 10 shapes per complexity, from the SCUT surface reconstruction benchmark [25].

	Simple			Ordinary			Complex		
Method	NCS	CD	Fs	NCS	CD	Fs	NCS	CD	Fs
WNNC (Ω_0)	0.989	0.279	0.879	0.970	**0.170**	0.984	0.949	0.190	0.969
WNNC (SD, A, G_B, Ω_0)	0.989	0.279	0.879	0.970	0.171	0.984	**0.952**	**0.164**	**0.979**

Table 6. Results for non-uniform data, 10 shapes per complexity, from the SCUT surface reconstruction benchmark [25].

	Low			Medium			High		
Method	NCS	CD	Fs	NCS	CD	Fs	NCS	CD	Fs
WNNC (Ω_0)	0.694	**0.423**	**0.722**	0.835	0.708	0.887	0.756	0.820	0.812
WNNC (SD, A, G_B, Ω_0)	**0.712**	0.443	0.716	**0.862**	**0.293**	**0.925**	**0.780**	**0.580**	**0.814**
WNNC (Ω_1)	0.804	**0.476**	**0.650**	0.934	**0.343**	**0.931**	0.912	0.345	0.835
WNNC (SD, A, G_B, Ω_1)	**0.808**	0.495	0.645	**0.935**	0.355	0.929	**0.918**	**0.334**	**0.836**

Table 7. Results under three noise levels (30 shapes at each level) from the SCUT surface reconstruction benchmark [25].

	Low			Medium			High		
Method	NCS	CD	Fs	NCS	CD	Fs	NCS	CD	Fs
WNNC (Ω_0)	0.931	1.737	0.834	0.924	**1.895**	**0.803**	0.905	**2.252**	**0.717**
WNNC (SD, A, G_B, Ω_0)	0.931	**1.735**	0.834	0.924	1.898	0.802	0.905	2.267	0.716
WNNC (Ω_1)	0.929	**2.162**	0.815	0.923	2.499	0.780	0.902	2.687	0.698
WNNC (SD, A, G_B, Ω_1)	0.929	2.168	0.815	0.923	**2.488**	0.780	0.902	**2.680**	0.698

Table 8. Results under three levels of missing data (30 shapes at each level) from the SCUT surface reconstruction benchmark [25].

D. Further Visualizations

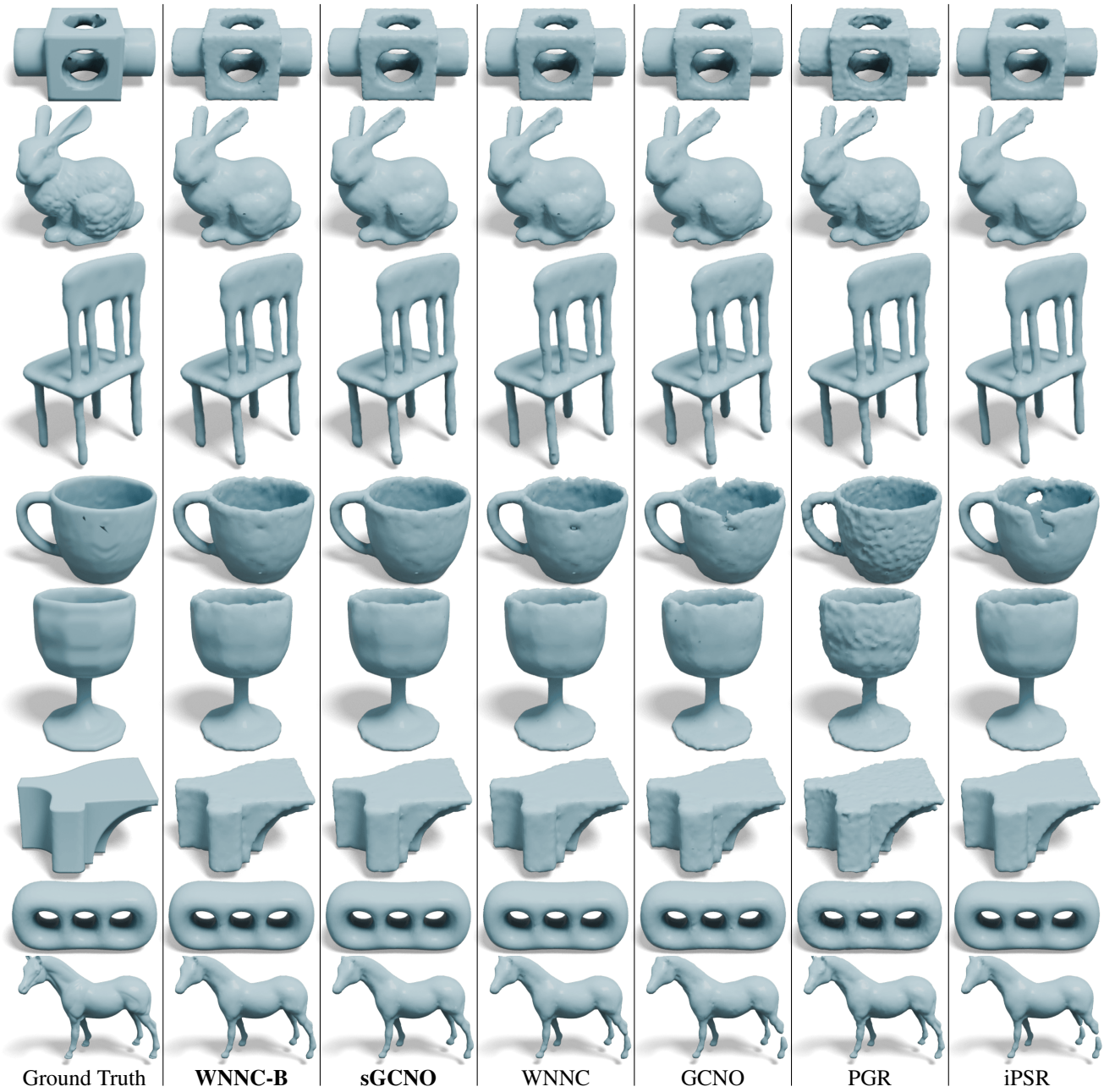

Figure 7. Comparison of normal estimation methods on GCNO's dataset. WNNC-B denotes our bounded formulation of WNNC (LM, A_B, G_B, Ω_1); sGCNO denotes our stochastic bounded formulation of GCNO; and WNNC denotes WNNC (Ω_1).

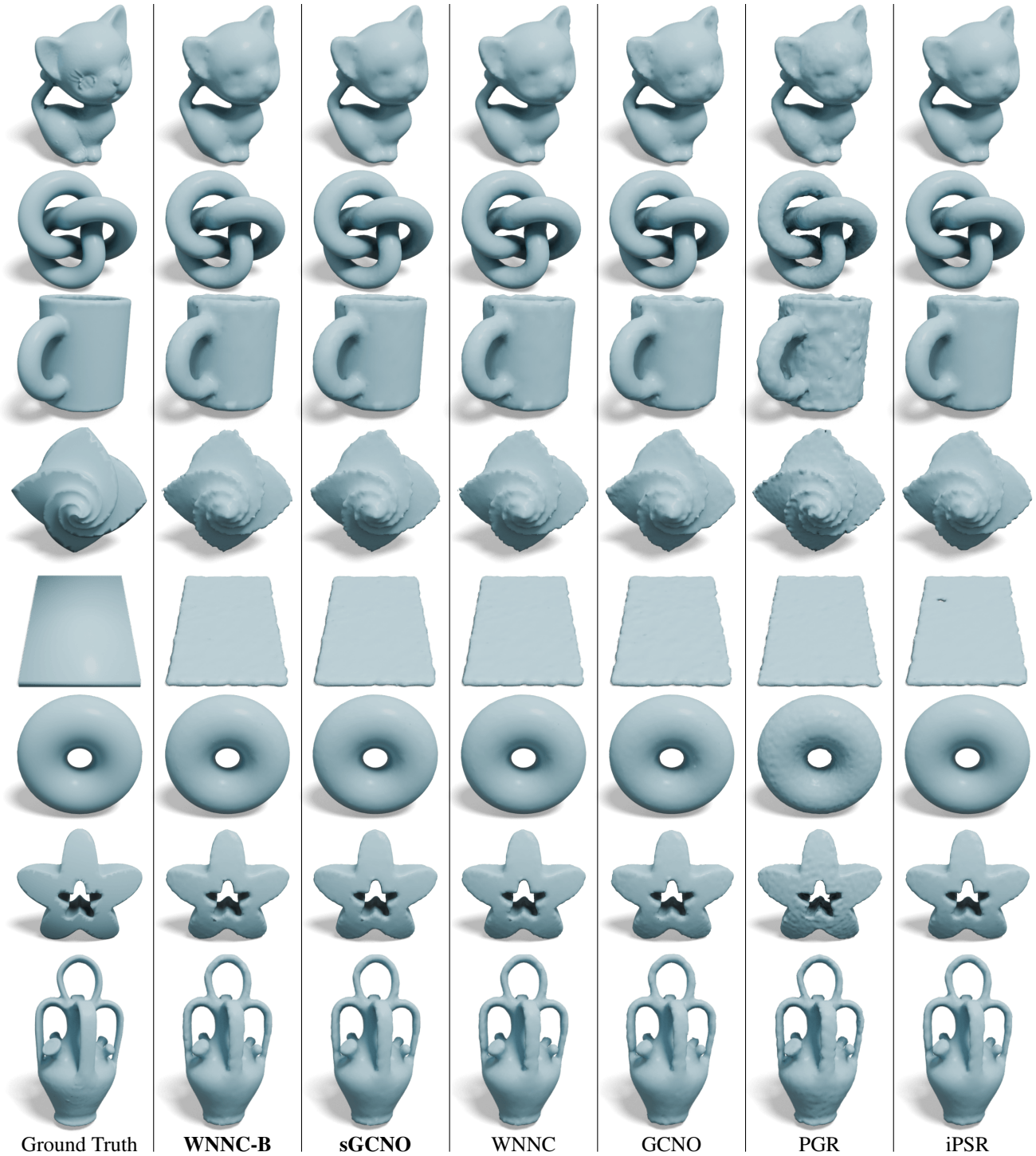

Figure 8. Comparison of normal estimation methods on GCNO's dataset. WNNC-B denotes our bounded formulation of WNNC (LM, A_B, G_B, Ω_1); sGCNO denotes our stochastic bounded formulation of GCNO; and WNNC denotes WNNC (Ω_1).

E. Further Details on GCNO

Xu *et al*. [54] propose an algorithm for normal estimation, called globally consistent normal orientation (GCNO), that regularizes the winding number field. Given an input point cloud $\mathcal{P} = \{p_i\}_{i=1}^N$, the algorithm first computes the 3D Voronoi diagram of $\mathcal{P}$. It then estimate areas for each point by the cross section of that point's Voronoi cell in the plane orthogonal to direction that the cell is longest. Normals are parameterized using spherical coordinates and are initialized randomly. The algorithm then optimizes these normals using L-BFGS with respect to a loss function that involves computing the winding number at each vertex of the Voronoi diagram.

Given the input point cloud $\mathcal{P} = \{p_i\}_{i=1}^N$ with current normal estimates n_i and fixed area estimates a_i, domain queries (all Voronoi vertices) $\mathcal{Q} = \{q_j\}_{j=1}^M$ and per-point queries (chosen as the Voronoi vertices defining the cell about each input point) $\mathcal{Q}_i = \{q_k^i\}_{k=1}^{M_i}$ $i = 1, .., N$, the loss function is formulated as the sum of three components,

$$\mathcal{L} = \frac{1}{N}\Big(\mathcal{L}_{01} + \lambda_B \mathcal{L}_B + \lambda_A \mathcal{L}_A\Big) \tag{55}$$

$$\mathcal{L}_{01} = \sum_{j=1}^{M} f_{\text{SDW}}(w(q_j)) \tag{56}$$

$$\mathcal{L}_B = -\sum_{i=1}^{N}\left(\frac{1}{M_i}\sum_{k=1}^{M_i}\left(w(q_k^i) - \bar{w}^i\right)^2\right) \tag{57}$$

$$\mathcal{L}_A = \sum_{i=1}^{N}\left(\frac{1}{M_i}\sum_{k=1}^{M_i} w(q_k^i) n_i^{\mathsf{T}}\left(q_k^i - p_i\right)^2\right), \tag{58}$$

where

$$f_{\text{SDW}}(x) = f_{\text{DW}} - \frac{x}{4} \tag{59}$$

$$f_{\text{DW}}(x) = 4\left(x - \tfrac{1}{2}\right)^4 - 2\left(x - \tfrac{1}{2}\right)^2 \tag{60}$$

$$\bar{w}^i = \frac{1}{M_i}\sum_{k=1}^{M_i} w(q_k^i). \tag{61}$$

This optimizes the normals n_i for three criteria on the winding number field. Component $\mathcal{L}_{01}$ optimizes for the winding number to be either zero or one at all domain queries q_j, as the ground truth winding number field for the surface should be zeor outside and one inside. The double well function $f_{\text{DW}}(x)$ and its sheared version $f_{\text{SDW}}(x)$ are shown in Fig. 9, the latter is used to encourage the value one more as they note that random normals makes the winding number usually around zero. Loss component $\mathcal{L}_B$ optimizes for the winding number at queries q_k^i around each input point p_i to be balanced with roughly equal number of zeros and ones, as p_i should be on a surface that separates inside from outside. In particular they maximize the variance of $w(q_k^i)$, as this tends to balanced winding numbers whose values are restricted to zero and one. Finally, $\mathcal{L}_A(\boldsymbol{n})$ optimizes for the normal at each point to align with the Voronoi vertices with the lowest winding number.

F. Further Details on Our sGCNO

F.1. Overview

Set up and area estimation The input point cloud is first translated and scaled to fit within $[-1, 1]^3$. Normals are parameterized using spherical coordinates and initialized randomly. We take an efficient and effective approach to estimate point areas, the area for each point is the area of a circle with radius given by the distance to its nearest neighbor. We hypothesize that this should be sufficient in most cases as the winding number field gracefully degrades with holes [26]. We find that these estimates are good on average, so the total point cloud area is reasonable though individual areas may deviate from the true area depending on the sampling. As a result, point areas are allowed to be optimized within some trust interval of the estimate (see Appendix F.2).

Sampling. At each iteration, samples are generated with respect to a subset of the input points $\{p_i\}_{i\in S}$ where $S \subset \{1, ..., N\}$. Around each point Gaussian noise is sampled with two different standard deviations, M_c coarse samples and M_f fine samples with variance σ_c and σ_{f_i}, respectively. We set σ_c to 0.3 and σ_{f_i} to the average distance from p_i to its four nearest

neighbors. For the term $\mathcal{L}_{01}$ we use both the coarse samples $\mathcal{Q}_{c,i} = \{q_k^{(c,i)}\}_{k=1}^{M_c}$ and the fine samples $\mathcal{Q}_{f,i} = \{q_k^{(f,i)}\}_{k=1}^{M_f}$ for all $i \in S$. However, we only use the fine samples for the balance term $\mathcal{L}_B$. This set up is quite flexible—if the input point cloud is quite dense then a small subset S can be used effectively, and the accuracy of $\mathcal{L}_B$ can be adjusted by varying M_f.

Loss functions. Our loss is comprised of two terms from Xu *et al*. [54], $\mathcal{L}_{01}$ and $\mathcal{L}_B$. We omit the third term, $\mathcal{L}_A$ as we do not compute a Voronoi diagram. While being a good heuristic it does not allow for accurate normals. We further improve on losses $\mathcal{L}_{01}$ and $\mathcal{L}_B$ in Appendix F.3.

As noted in GCNO, $\mathcal{L}_{01}$ and $\mathcal{L}_B$ are good at orienting normals in the right direction (*i.e*., facing outward), but do not give good angular estimates. As such, we propose two additional losses, an alignment loss to make the normals consistent with its surrounding winding number field (essentially smoothing it) and a surface loss to ensure that the input points lie on the 0.5 level set. See in Appendix F.3.

Scaling for large point clouds. Due to our sampling, in each batch we need to compute the winding number for $(M_c + M_f)|S|$ query points. Since each winding number requires a term from each input point, this means forming a matrix of size $(M_c + M_f)|S| \times N$. For large N, this may not fit within GPU memory. Note however that we are just summing over these matrices, not solving a matrix equation. Thus, we use the KeOps library [13, 14] which allows computing reductions of large arrays on the GPU without memory overflow. To do this, they do not actually form the array and instead have the user specify the large array and its reduction symbolically, which KeOPS then parallelizes on the GPU efficiently with linear (not quadratic) memory.

F.2. Optimizing point areas

We consider the point areas a_i as parameters of the optimization (alongside the normals). However, to avoid straying too far from our initially estimated areas a_i^{est}, we employ two mechanisms:: First, we specify boundary constraints such that $a_i \in (a_i^{\text{est}}/4, 1000a_i^{\text{est}})$, and, second, we add a loss that encourages the total area to remain close to the total area at initialization,

$$\mathcal{L}_{\text{area}} = \left| \left(\sum_{i=1}^{n} a_i \right) - \left(\sum_{i=1}^{n} a_i^{\text{est}} \right) \right|. \tag{62}$$

We find the latter to be quite important so that the areas do not change too rapidly at the start of optimization.

F.3. Loss functions

01 loss. An issue with GCNO's sheared double well, Eq. (59), is that its minima are not actually at zero and one (see Fig. 9). We formulate the functions that have this property by integrating the required derivative, which gives

$$f_{DW}(x; a, c) = \frac{a}{4}x^4 - \left(\frac{a}{3} + \frac{ac}{3}\right)x^3 + \frac{1}{2}acx^2. \tag{63}$$

When $a = 16$ and $c = 0.5$ we get back the symmetric double well Eq. (60) with some vertical translation. To shear, we can change c to be closer to zero. In particular, we find that $c = 0.41$ resembles Eq. (56) quite well. On the other hand, we can change how much the loss penalizes by changing a, and we find that using $a = 8$ works better.

We use the sheared version of Eq. (63) for coarse samples, also observing GCNO's findings that the distribution of the ground truth $w(q)$ for coarse samples is highly skewed towards one. However we note that fine samples the distribution is balanced, so for those samples we use our symmetric loss. Thus our loss is

$$\mathcal{L}_{01} = \left(\frac{1}{2|S|} \sum_{i \in S} \sum_{k=1}^{M_c} f_{DW}\left(w\left(q_k^{c,i} \right); 8, 0.41 \right) \right) + \left(\frac{1}{2|S|} \sum_{i \in S} \sum_{k=1}^{M_f} f_{DW}\left(w\left(q_k^{f,i} \right); 8, 0.5 \right) \right). \tag{64}$$

Balance loss. The aim of the balance loss is to enforce that $w(q) = 0$ and $w(q) = 1$ is evenly distributed around each input point. We find that with our sampling, purely maximizing variance (as per Eq. (57)) leads to increasing the number of winding numbers that are outside of $[0, 1]$. Thus we instead note what the variance should be for an equal number of zeros

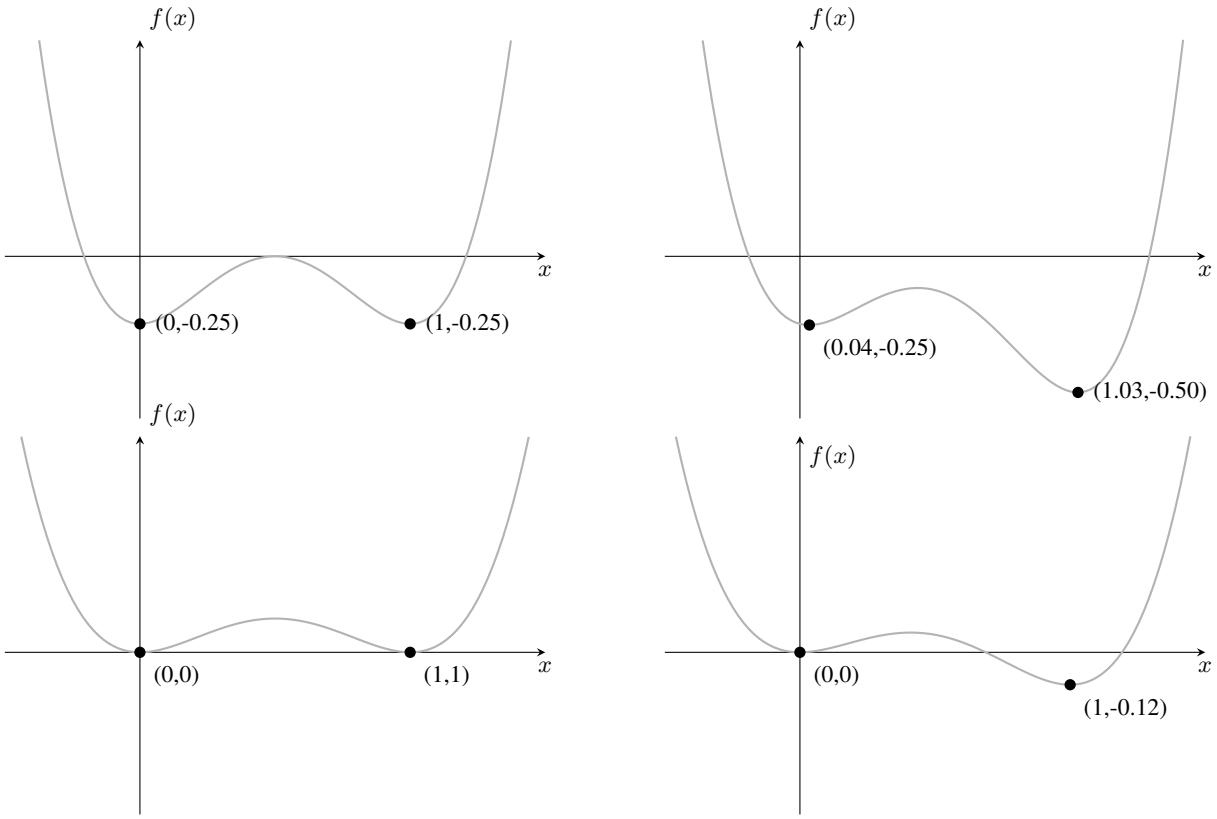

Figure 9. GCNO's double well functions $f_{\text{DW}}(x)$ (**top left**), $f_{\text{SDW}}(x)$ (**top right**) and our double well functions $f_{\text{DW}}(x; 8, 0.5)$ (**bottom left**), $f_{\text{DW}}(x; 8, 0.41)$ (**bottom right**). Note that $f_{\text{SDW}}(x)$ minima are not exactly at zero and one, and our functions penalize less.

and ones and constrain the variance to reach that

$$\mathcal{L}_{\text{var}} = \frac{1}{|S|} \sum_{i \in S} \left| \left(\frac{1}{M_f - 1} \sum_{k=1}^{M_f} \left(w\left(q_k^{f,i}\right) - \bar{w}^i \right)^2 \right) - \sigma_{\text{opt}}^2 \right| \tag{65}$$

where $\bar{w}^{f,i}$ is the mean winding number and the target variance is calculated as $\sigma_{\text{opt}}^2 = \frac{1}{M_f - 1} \sum_{k=1}^{M_f} 0.5^2 = \frac{M_f}{4(M_f - 1)}$. In the SDF literature it is common to do this kind of balance loss by constraining the mean of the samples rather than the variance [37]. We find that this helps initial optimization more than the variance loss. Thus the mean loss is

$$\mathcal{L}_{\text{mean}} = \frac{1}{|S|} \sum_{i \in S} \left(\left(\frac{1}{M_f} \sum_{k=1}^{M_f} w\left(q_k^{f,i}\right) \right) - \frac{1}{2} \right)^2 \tag{66}$$

and our balance loss is

$$\mathcal{L}_B = \mathcal{L}_{\text{mean}} + \mathcal{L}_{\text{var}}. \tag{67}$$

Alignment loss. Successfully optimizing the winding number ensures that the normals are oriented to be consistent with the input points bounding an interior region (in the sense that they point in the same direction), however there is no guarantee that the normals actually align well with the normals of that interior region. However, if the normals are consistent then the average winding number field around each point should align with the ground truth normals. Thus using our fine samples we compute the gradient of the winding number field (analytically) around each point and average it to get a rough inward

normal direction, and constrain our normals to align with that

$$\mathcal{L}_{\text{align}} = \frac{1}{|S|} \sum_{i \in S} \left(1 - \left\langle n_i^{\text{est}}, n_i \right\rangle\right) \tag{68}$$

$$n_i^{\text{est}} = -\left(\frac{1}{M_f} \sum_{k=1}^{M_f} \widehat{\partial_q w \left(q_k^{f,i}\right)} \right). \tag{69}$$

Surface loss. Ideally the surface of the shape should be on the 0.5 level set of the winding number. We add a loss to enforce that the input points lie on this level set:

$$\mathcal{L}_S = \frac{1}{|S|} \sum_{i \in S} \left| w(p_i) - \frac{1}{2} \right|. \tag{70}$$

We appropriately add small values to any denominator of the form $\|p_i - q\|_2$ in Eq. (12) so that computing $w(q)$ at input points is well defined.