Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Learned Bias Correction for Surface-Area Estimation from CT-like Voxel Data

Manas Jain
Indian Institute of Technology Roorkee
Mayank Goswami
mayankfph@iitr.ac.in
Indian Institute of Technology Roorkee

Research Article

Keywords: Computed tomography, surface area, marching cubes, partial-volume effect, deep learning, 3D CNN, PointNet, synthetic data

Posted Date: November 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11685202/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Learned Bias Correction for Surface-Area Estimation from CT-like Voxel Data

Manas Jain[1] and Mayank Goswami,[1,2]

[1]Department of Physics, IIT Roorkee
[2]MF School of AI and DS, IIT Roorkee
mayank.goswami@ph.iitr.ac.in

***Abstract*—Measuring the surface area of an object from a CT scan is harder than it looks, because the scan is a grid of small cubes (voxels) and the true surface passes between them. Counting the exposed faces of the voxels gives an area about 50% too large at every resolution. Marching cubes, which fits a triangle mesh to the data, is closer, but its error depends on how curved the object is and how noisy the scan is. We built 2,400 synthetic shapes from seven families whose true surface area is known to better than 0.01%, and turned them into 14,400 CT-like volumes at 32^3 and 64^3 voxels with three noise levels. We then trained models to predict, for each object, how much the marching-cubes area should be corrected: a Random Forest on ten simple shape measurements, a 3D convolutional neural network (CNN) and a PointNet. On test shapes that none of the models had seen, marching cubes has an average error of 4.78% (32^3) and 3.52% (64^3). The best corrections bring this down to 0.68% (3D CNN, 32^3) and 0.18% (Random Forest, 64^3), and stay below 2% on two shape families left out of training altogether. The simple Random Forest does as well as the CNN overall, the CNN is better only for objects smaller than 16 voxels, and no method can recover structures thinner than about two voxels.**

***Index Terms*—Computed tomography, surface area, marching cubes, partial-volume effect, deep learning, 3D CNN, PointNet, synthetic data.**

I. Introduction

Surface area is one of the most requested measurements in X-ray computed tomography (CT) of materials and parts. It controls reaction rates in catalysts, flow in porous media and the quality of additively manufactured lattices. It is usually measured from the reconstructed voxel volume, either by counting the exposed faces of segmented voxels or by extracting a triangle mesh with marching cubes [1] and summing the triangle areas.

Both routes are biased. Counting voxel faces measures the area of a staircase, which is about 50% larger than the true surface at every resolution. Marching cubes on a thresholded (binary) volume is better, but its error does not vanish as the voxels get smaller [2, 3]. Marching cubes on the greyscale volume is accurate for clean, well-resolved data, but real CT volumes are noisy and many structures are only a few voxels across. Because the error depends on resolution, noise, curvature and thin features at the same time, a single correction factor cannot remove it.

This work builds a synthetic CT-like dataset with exactly known surface areas and trains models that predict a correction to the greyscale marching-cubes area for each object, so that learning acts as a correction layer on top of the classical measurement rather than replacing it.

Corresponding author: mayank.goswami@ph.iitr.ac.in

II. Problem, Objectives and Motivation

A. Problem

Given a CT-like greyscale volume of a single object on an N^3 grid, estimate the area of the object's true, continuous surface. The volume is degraded by the finite voxel size (partial-volume averaging), by the scanner blur and by noise.

B. Objectives

The objectives of this project are:

1) Build shapes with reference areas accurate to better than 10^{-4}, so that the reference adds no error of its own.
2) Measure the error of voxel-face counting and three marching-cubes variants across resolution and noise.
3) Train a 3D CNN and a PointNet to correct the greyscale marching-cubes area, and compare them with a per-regime lookup table and a Random Forest.
4) Test the corrections on shape families never seen in training.
5) State when a learned correction is worth using.

C. Motivation

A 5% error in surface area is large compared with the effects such measurements are used to detect, for example differences between catalyst batches or the deviation of a printed lattice from its design. Errors of this size are typical for small or noisy objects (Section VI). A correction learned once on synthetic data is cheap to apply to every new object.

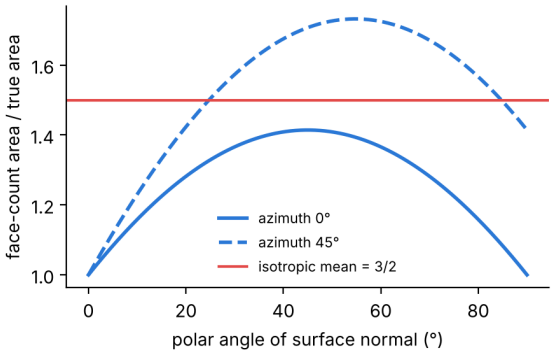


Fig. 1. Face-count area per unit true area, $|n_x| + |n_y| + |n_z|$, for a plane whose normal makes the given polar angle with the z axis. The average over all orientations is 3/2.

III. Theory

In short: counting voxel faces measures a staircase, so it is always too high; marching cubes on a black-and-white volume can only build a few triangle directions, so it keeps a small zig-zag error; and marching cubes on the greyscale volume is exact on flat surfaces but slightly too small on curved ones and too large on noisy ones.

A. Voxel-face counting

A flat piece of surface with unit normal $\mathbf{n} = (n_x, n_y, n_z)$ becomes a staircase of voxel faces. The faces normal to each axis add up to the projection of the true area on that plane, so the face-count area per unit true area is

$$r(\mathbf{n}) = |n_x| + |n_y| + |n_z|, \qquad 1 \le r \le \sqrt{3}. \qquad (1)$$

It equals 1 only for surfaces aligned with the grid. For random orientations each $|n_i|$ averages 1/2, so r averages 3/2 (Fig. 1): a 50% overestimate that does not depend on the voxel size. On a clean sphere at 128^3 we measure +50.03%.

B. Marching cubes on binary and greyscale volumes

On a binary volume, marching cubes places every vertex at an edge midpoint, so only a few triangle orientations occur and a tilted plane becomes a fine zig-zag. Smaller voxels make the zig-zag finer but not flatter, so the error does not go to zero [2, 3]: on clean spheres we measure +7.3% at 16^3 and +8.8% at 128^3.

On a greyscale volume the vertices are interpolated from the partial-volume intensities, so a flat surface is recovered exactly. A curved surface is not: blurring spreads the object's intensity outwards over a larger volume, so the half-intensity surface ends up slightly inside the true one. For a sphere of radius R voxels the error is about $-2\sigma^2/R^2$, where $\sigma^2 \approx 0.44$ voxel^2 is the total blur. This predicts −0.56% for a sphere on a 32^3 grid; we measure −0.62%. Extra smoothing makes the error about three times bigger (Appendix A).

C. Effect of noise

Noise with standard deviation σ_n adds area in two ways. It roughens the surface, and a rough surface always has more area than a smooth one. And background voxels cross the iso-level by chance with probability $p = Q(0.5/\sigma_n)$, each becoming a tiny spurious surface. At $\sigma_n = 0.15$, $p \approx 4.3 \times 10^{-4}$ predicts about 112 such specks on a 64^3 grid; the measured Euler numbers, which count them, are about 100 to 125. Smoothing removes the specks but shrinks curved surfaces and erases structures thinner than about two voxels. Greyscale marching cubes therefore has a negative, curvature-driven error and a positive, noise-driven error whose sizes depend on the object and the scan, which is why a single correction factor cannot fix both.

D. Reference area and learning target

Spheres, cylinders, tori and ellipsoids have exact area formulas. Random blobs and superellipsoids are measured on very fine triangle meshes at two levels of detail, combined to cancel the leading error term; the remaining error is below 0.01% for every shape (Appendix B). All models predict $y = \log(A_{\text{true}}/A_{\text{MC}})$, where A_{MC} is the greyscale marching-cubes area, and the estimate is $\hat{A} = A_{\text{MC}}\, e^{\hat{y}}$. This target does not depend on the object's size or orientation, so each training volume can be augmented exactly with the 48 symmetries of the cube.

IV. Previous Work

Marching cubes [1] is the standard way to extract a surface from a voxel volume; we use the variant of Lewiner et al. [4] in scikit-image [5]. Nyström et al. [6] found triangulated surfaces considerably more accurate than voxel-face surfaces. In digital geometry, Mullikin and Verbeek [7], Windreich et al. [8] and Lindblad [9] derived optimal weights for local $2 \times 2 \times 2$ voxel configurations. On convergence, Kenmochi and Klette [2] showed that binary marching cubes does not converge to the true area even for spheres, and Svane [3] proved that no local estimator converges on convex polytopes (see also Ziegel and Kiderlen [10], Klette and Rosenfeld [11]). On the learning side, 3D CNNs [12] and PointNet [13] are established tools, and Rabbani et al. [14] predicted properties of porous media from micro-CT with a CNN, without an exact ground truth for surface area. We did not find a study that measures, against an exact reference, how much a learned model reduces the surface-area error of marching cubes, separately by resolution, noise and unseen shape family, and compares it with a simple feature-based baseline [15].

V. Method

A. Data

Table I lists the seven shape families. Five are used for training: spheres (150 shapes), ellipsoids (350), capped

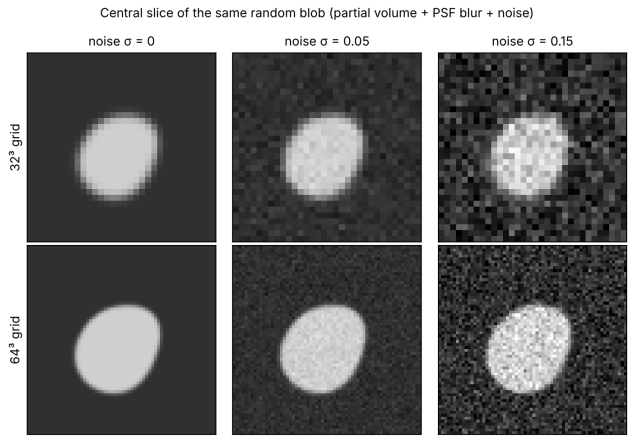


Fig. 2. Central slice through the same random blob at both resolutions and all three noise levels.

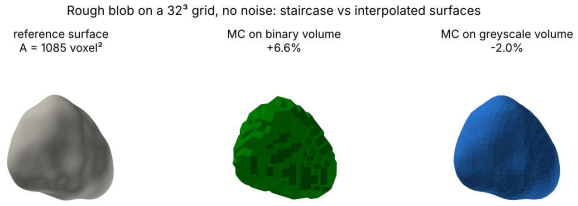


Fig. 3. Reference surface of a rough blob and the marching-cubes meshes from its clean 32^3 volume. The binary mesh is a staircase; the greyscale mesh is smooth but loses the finest detail.

cylinders (300), tori (300) and smooth random blobs (900), the last being star-shaped surfaces built from six low-frequency waves on the sphere. Two families are held out for testing only: superellipsoids (200), with pointed tips or box-like rounded edges, and rough blobs (200), which add ten high-frequency waves. Each shape is scaled to fill 35–100% of the grid (diameters 9.8 to 60 voxels), randomly rotated and shifted by a sub-voxel offset. The volume is formed by a simple CT model: partial-volume occupancy computed with $5\times5\times5$ supersampling, a Gaussian blur with $\sigma = 0.6$ voxel, and Gaussian noise with $\sigma_n \in \{0, 0.05, 0.15\}$ relative to unit contrast, stored as 8-bit integers. Every shape is rendered at $N = 32$ and 64 and all three noise levels, giving six volumes per shape and 14,400 in total. The split is by shape (70/15/15 within each training family, fixed seed 51), so all six volumes of a shape stay together: 1,400 training, 299 validation and 301 test shapes, plus the 400 held-out shapes. Figure 2 shows one shape at both resolutions and all noise levels.

B. Estimators

Classical: voxel-face counting after thresholding at 0.5; marching cubes on the thresholded (binary) volume; marching cubes on the greyscale volume at iso-level 0.5; and marching cubes after Gaussian smoothing with $\sigma_s = 1$ voxel (Fig. 3). **MC + lookup** multiplies the greyscale area by the median training correction for the same resolution and noise level; it is given the true noise level, so any learned model has to beat it to show that it does more than apply an average. **Random Forest** [15] (400 trees) predicts y from ten simple measurements: the greyscale area relative to N^2, the ratios of the face, binary and smoothed areas to it, a compactness term, two aspect ratios, the Euler number of the thresholded volume, a noise estimate and N.

3D CNN: five convolutional blocks with 16 to 256 channels, batch normalisation, ReLU and max pooling, then global average and max pooling and a 128-unit head (3.59M parameters). **PointNet** [13]: 2,048 points sampled on the greyscale mesh, a shared MLP up to 1,024 features, max pooling and a 512–128 head (0.74M parameters), with random rotations instead of the alignment network. Both are trained per resolution with a Huber loss on $y/0.05$, AdamW, a one-cycle learning-rate schedule [16] with peak 10^{-3}, batch size 32, up to 40 epochs (15 for the CNN at 64^3) and early stopping on the validation error, using PyTorch [17] on an Apple-silicon laptop. We report the mean absolute percentage error (MAPE) and the mean signed error (bias) of $\hat{A}/A_{\text{true}} - 1$ on held-out shapes only.

VI. Results

All results are on shapes not used for training or model selection: the test split (301 shapes, 903 volumes per resolution) and the two held-out families (200 shapes each). Table II gives the overall picture.

A. Classical estimators

Voxel-face counting overestimates the area by 57% at 32^3 and 66% at 64^3, more than the 50% of the theory because noise adds isolated voxels whose faces also count. Binary marching cubes has a positive bias (+9.6% at 32^3, +14.5% at 64^3) that grows with resolution instead of shrinking. The resolution sweep (Fig. 4) shows why: on clean data the bias rises from +0.5% at 24^3 to +8.5% at 128^3, as the theory predicts, and at $\sigma_n = 0.15$ from +16% to +34%, because more voxels means more noise specks. Greyscale marching cubes is too small for small objects (−22.9% at 16^3 on clean data, from curvature) and too large at high noise (+8.6% at 64^3). Because these effects have opposite signs, its error is not smallest on clean data. Smoothing removes most of the noise error at 64^3 but costs 13.8% at 32^3, where most objects are small.

B. Learned and calibrated corrections

Every correction improves on raw greyscale marching cubes by a large factor. The lookup table already reduces the test MAPE from 4.78% to 2.97% at 32^3 and from 3.52% to 0.79% at 64^3, so most of the error

TABLE I

SHAPE FAMILIES. PARAMETERS ARE DRAWN UNIFORMLY FROM THE GIVEN RANGES; ALL SHAPES ARE NORMALISED TO MAXIMUM EXTENT 1 BEFORE PLACEMENT.

Family	Role	Shapes	Parameters
Sphere	train	150	radius 1
Ellipsoid	train	350	$a = 1$, $b, c \in [0.35, 1]$
Cylinder (capped)	train	300	radius $\in [0.3, 0.85]$, half-height $\sqrt{1-\rho^2}$
Torus	train	300	$r/R \in [0.2, 0.6]$, $R + r = 1$
Smooth blob	train	900	6 waves, $\omega \in [1, 3]$, s.d. of $\log r \in [0.15, 0.30]$
Superellipsoid	held out	200	$b, c \in [0.45, 1]$; $p \in [1.5, 1.8]$ (40%) or $[3, 6]$ (60%)
Rough blob	held out	200	smooth blob + 10 waves, $\omega \in [8, 14]$, s.d. $\in [0.02, 0.05]$

TABLE II

SURFACE-AREA ERROR (%) ON HELD-OUT SHAPES, ALL NOISE LEVELS POOLED. MAPE: MEAN ABSOLUTE ERROR; BIAS: MEAN SIGNED ERROR. THE TWO RIGHT-HAND BLOCKS ARE SHAPE FAMILIES NEVER SEEN IN TRAINING (MAPE). BEST VALUE PER COLUMN IN BOLD.

	Test MAPE		Test bias		Superellipsoid		Rough blob	
Method	32^3	64^3	32^3	64^3	32^3	64^3	32^3	64^3
Voxel faces	56.71	65.54	+56.64	+65.54	57.45	64.81	55.30	65.65
MC (binary)	10.38	14.49	+9.56	+14.49	10.51	14.31	8.91	13.99
MC (greyscale)	4.78	3.52	−1.25	+2.41	4.10	3.38	5.27	3.57
MC (smoothed)	13.78	3.04	−13.78	−3.04	9.93	2.29	14.02	4.37
MC + lookup	2.97	0.79	−1.45	−0.46	1.87	0.43	3.22	1.09
Random Forest	0.72	**0.18**	−0.18	**+0.04**	**0.73**	**0.39**	**1.71**	**0.48**
PointNet	1.98	1.04	+0.33	+0.34	1.54	0.95	3.00	1.32
3D CNN	**0.68**	0.26	**−0.10**	+0.10	1.05	0.45	1.96	0.55

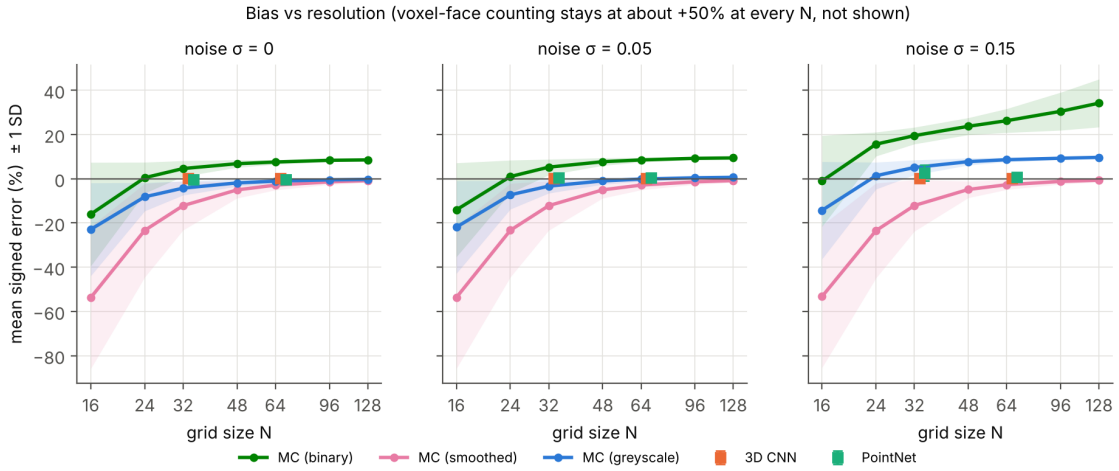


Fig. 4. Mean signed error (±1 s.d.) of the marching-cubes estimators against grid size for 48 test shapes. Learned models are shown at the two sizes they were trained on.

at 64^3 is a systematic, regime-dependent bias, while at 32^3 it depends on the individual object. The 3D CNN reaches 0.68% and 0.26%, a seven-fold and a thirteen-fold reduction, with a bias within ±0.1%. The Random Forest performs almost identically at 32^3 (0.72%) and is the best model at 64^3 (0.18%). Its most important inputs are the Euler number (0.29), the noise estimate (0.26), the smoothed-to-raw marching-cubes area ratio (0.22) and the face-count-to-marching-cubes area ratio (0.08). The first two measure exactly the two noise effects of Section III. PointNet is clearly weaker (1.98% and 1.04%), because sampling points from a mesh that already contains the noise artefacts throws away the intensity information the volume-based models use.

C. Noise and object size

The learned models are nearly insensitive to noise (Table III): at 64^3 the CNN's MAPE goes from 0.20% on clean data to 0.40% at $\sigma_n = 0.15$, while greyscale marching cubes goes from 1.13% to 8.56%. Object size

TABLE III
TEST MAPE (%) BY RESOLUTION AND NOISE LEVEL. BEST PER COLUMN IN BOLD.

	32^3			64^3		
Method	σ_n=0	0.05	0.15	0	0.05	0.15
Voxel faces	44.10	45.64	80.38	48.31	49.91	98.40
MC (binary)	5.24	5.85	20.05	7.64	8.49	27.34
MC (greyscale)	4.73	3.84	5.77	1.13	0.86	8.56
MC (smoothed)	13.87	13.82	13.65	3.11	3.09	2.93
MC + lookup	3.06	3.05	2.80	0.84	0.83	0.71
Random Forest	0.49	0.61	1.07	**0.09**	**0.14**	**0.30**
PointNet	1.42	1.41	3.13	0.75	0.69	1.67
3D CNN	**0.46**	**0.55**	**1.02**	0.20	0.19	0.40

TABLE IV
TEST MAPE (%) BY OBJECT DIAMETER IN VOXELS (BOTH RESOLUTIONS POOLED).

Method	< 16	16–24	24–32	32–48	48–64
MC (greyscale)	6.43	3.89	3.57	3.45	3.44
MC (smoothed)	24.52	8.08	4.84	2.53	1.45
MC + lookup	4.76	1.78	1.48	0.67	0.58
Random Forest	1.41	**0.39**	**0.22**	**0.13**	**0.08**
PointNet	2.14	1.57	1.54	0.81	1.48
3D CNN	**1.24**	0.39	0.28	0.22	0.22

matters more than grid size. For objects less than 16 voxels across, greyscale marching cubes has a MAPE of 6.43%, the Random Forest 1.41% and the CNN 1.24%, the size class where the CNN has its clearest advantage. Above 48 voxels the Random Forest reaches 0.08% (Table IV).

D. Unseen shape families and a failure case

Both held-out families stay far below the classical errors (Fig. 5). On superellipsoids, whose sharp edges do not occur in training, the CNN reaches 1.05% and 0.45% and the Random Forest 0.73% and 0.39%, against 4.10% and 3.38% for greyscale marching cubes. Rough blobs are the hardest case (CNN 1.96%, Random Forest 1.71% at 32^3), with corrections biased low: the models were trained on smooth shapes and cannot add back area hidden in sub-voxel roughness. At 64^3 the error falls to about 0.5%. Table V breaks the results down by family.

The error distributions are heavy-tailed at 32^3. The worst case is a test torus whose tube is 1.77 voxels thick: after blurring most of the tube stays below the iso-level, so greyscale marching cubes recovers only 42 of its 142 voxel^2, and every model is off by 47% to 66%. A correction of the marching-cubes area cannot recover a surface that is missing from the mesh.

E. Which estimator to use

Table VI summarises when a learned correction is worth the effort. Its benefit is largest where the classical methods are weakest: small objects, coarse grids and noisy data.

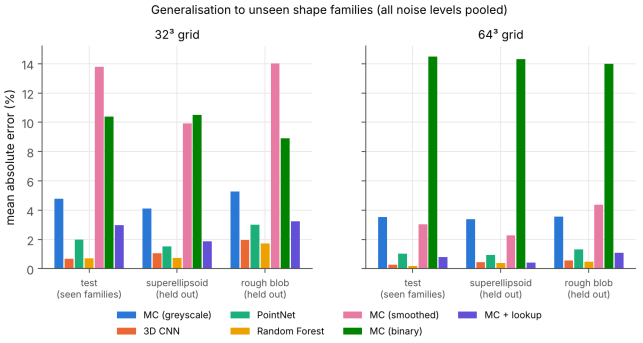


Fig. 5. MAPE on the test split and on the two held-out families (all noise levels pooled).

VII. CONCLUSION

The errors of the classical surface-area methods are not random, and that is what makes them correctable. Voxel-face counting is always about 50% too high, binary marching cubes keeps an error that does not shrink with resolution, and greyscale marching cubes is pulled down on curved surfaces, by an amount a simple blurring argument predicts, and pushed up by noise.

A correction learned from synthetic shapes with known area removes most of this error. On test shapes it cuts the error of greyscale marching cubes from 4.78% to 0.68% at 32^3 and from 3.52% to 0.18% at 64^3, barely changes with noise, and stays below 2% on shape families never seen in training. A Random Forest on ten simple measurements does as well as a 3D CNN with 3.6 million parameters; the CNN is worth its extra cost mainly for small objects on coarse grids. Structures thinner than about two voxels and roughness smaller than one voxel remain out of reach, because the information is no longer in the image. Natural next steps are a more realistic CT simulation, volumes with many touching structures, and a test on a real scan of an object with a known surface area.

VIII. LIMITATIONS

- **Simplified physics.** Noise is white and added in the image domain; real CT noise is correlated and signal dependent, and real volumes have beam-hardening, ring and streak artefacts.
- **One isolated object per volume.** Porous media and lattices with many touching structures are untested.
- **Fixed acquisition.** Two grid sizes, three noise levels and a fixed blur; behaviour outside this range is untested.
- **Sub-voxel roughness.** On rough surfaces the correction is a learned prior, not a measurement.
- **Limited shapes, no tuning.** Training shapes are star-shaped or of genus one; there was no hyperparameter tuning, each model was trained once, and the 64^3 CNN only for 15 epochs.

TABLE V
MAPE (%) BY SHAPE FAMILY (TEST SPLIT AND HELD-OUT FAMILIES, ALL NOISE LEVELS). †HELD OUT FROM TRAINING. BEST PER ROW AND RESOLUTION IN BOLD.

	32^3					64^3				
Family	MC grey	Lookup	RF	PointNet	3D CNN	MC grey	Lookup	RF	PointNet	3D CNN
Sphere	3.46	1.98	**0.19**	2.21	0.23	3.41	0.39	**0.06**	1.14	0.13
Ellipsoid	4.22	2.36	0.52	2.00	**0.45**	3.45	0.48	**0.20**	1.20	0.28
Cylinder	5.99	4.75	1.15	2.60	**0.78**	3.93	2.42	**0.31**	1.58	0.37
Torus	6.89	5.26	1.90	2.83	**1.85**	3.53	0.87	**0.26**	1.00	0.28
Smooth blob	4.12	2.03	**0.35**	1.45	0.41	3.43	0.41	**0.11**	0.79	0.23
Superellipsoid†	4.10	1.87	**0.73**	1.54	1.05	3.38	0.43	**0.39**	0.95	0.45
Rough blob†	5.27	3.22	**1.71**	3.00	1.96	3.57	1.09	**0.48**	1.32	0.55

TABLE VI
RECOMMENDED ESTIMATOR BY REGIME, BASED ON THE TEST RESULTS.

Regime	Recommendation
Any case	Do not use voxel-face counting or binary marching cubes for absolute area.
Clean data, large objects	Greyscale marching cubes is within about 1% at 64^3.
Noisy data ($\sigma_n = 0.15$)	A learned correction (CNN or Random Forest) gives 0.3–1.1%.
Small objects (< 16 voxels)	Use the 3D CNN (1.24%, against 6.43% for marching cubes).
Thinner than about 2 voxels	No method tested here is reliable.

- **No real scan.** No measurement on a real CT scan with a known surface area is included.

Code and Data Availability

Code, trained weights and result tables are in the project repository; the dataset is fixed by its generation script and seed (51) and is shared alongside the code.

IX. Video

https://youtu.be/gI7sTCV-atg

Appendix A
Curvature bias of greyscale marching cubes

For a plane and a symmetric blur the 0.5 iso-surface of the blurred image coincides with the true surface. For a curved surface it does not. If the effective blur is Gaussian with variance σ_{eff}^2, the level sets of the blurred indicator move towards the centre of curvature by

$$\delta \approx \sigma_{\text{eff}}^2 H, \qquad H = \tfrac{1}{2}(\kappa_1 + \kappa_2), \tag{2}$$

where H is the mean curvature. The intuition is that blurring a curved object spreads its intensity outwards over a larger volume, so the half-intensity surface ends up slightly inside the true one, and more so where the surface bends sharply. (Formally this follows from the heat equation, since Gaussian blurring is diffusion for a time $\sigma_{\text{eff}}^2/2$.) For a sphere of radius R this gives a relative area error of $-2\delta/R = -2\sigma_{\text{eff}}^2/R^2$. In our forward model the voxel box ($\sigma^2 = 1/12$) and the Gaussian PSF ($\sigma = 0.6$) combine to $\sigma_{\text{eff}}^2 \approx 0.443$ voxel^2; adding a smoothing filter with $\sigma_s = 1$ raises it to 1.443. Table VII compares this prediction with the measured errors on clean spheres. The agreement is close at low resolution, where the curvature term dominates; at 64^3 the remaining discrepancy is of the same size as the interpolation error of marching cubes itself.

TABLE VII
GREYSCALE MARCHING-CUBES ERROR ON CLEAN SPHERES: MEASURED VS THE CURVATURE PREDICTION $-2\sigma_{\text{EFF}}^2/R^2$ FROM EQ. (2). R IN VOXELS.

		MC on greyscale		MC after smoothing ($\sigma_s = 1$)	
Grid	R	predicted	measured	predicted	measured
16^3	5.4	−3.04%	−3.65%	−9.90%	−10.99%
32^3	12.6	−0.56%	−0.62%	−1.82%	−2.00%
64^3	27.0	−0.12%	−0.05%	−0.40%	−0.43%
128^3	55.8	−0.03%	+0.07%	−0.09%	−0.10%

Appendix B
Reference surface areas

Spheres, cylinders and tori have closed-form areas. For an ellipsoid with semi-axes $a \geq b \geq c$,

$$A = 2\pi c^2 + \frac{2\pi ab}{\sin\varphi}\left[E(\varphi, m)\sin^2\varphi + F(\varphi, m)\cos^2\varphi\right],$$
$$\cos\varphi = \frac{c}{a}, \qquad m = \frac{a^2(b^2 - c^2)}{b^2(a^2 - c^2)}, \tag{3}$$

with F and E the incomplete elliptic integrals of the first and second kind. The remaining families (random blobs and superellipsoids) are star-shaped, i.e. their surface is $\{r(\mathbf{u})\,\mathbf{u} : |\mathbf{u}| = 1\}$ for a radial function r. We triangulate them by mapping the vertices of a subdivided icosahedron to radius $r(\mathbf{u})$. For a smooth surface the area of such an inscribed mesh converges as $A_\ell = A + c\,h_\ell^2 + o(h_\ell^2)$

in the edge length h_ℓ, and each subdivision halves h_ℓ, so Richardson extrapolation of two levels,

$$A_{\text{ref}} = \tfrac{1}{3}\left(4A_{\ell+1} - A_\ell\right), \tag{4}$$

cancels the leading error term. We use levels 6 and 7 (81,920 and 327,680 triangles) and keep $|A_{\text{ref}} - A_7|$ as an error estimate for each shape.

References

[1] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Computer Graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[2] Y. Kenmochi and R. Klette, "Surface area estimation for digitized regular solids," in *Vision Geometry IX, Proceedings of SPIE*, vol. 4117, 2000, pp. 100–111.

[3] A. M. Svane, "On multigrid convergence of local algorithms for intrinsic volumes," arXiv preprint arXiv:1310.1783, 2013.

[4] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares, "Efficient implementation of marching cubes' cases with topological guarantees," *Journal of Graphics Tools*, vol. 8, no. 2, pp. 1–15, 2003.

[5] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, "scikit-image: image processing in Python," *PeerJ*, vol. 2, p. e453, 2014.

[6] I. Nyström, J. K. Udupa, G. J. Grevera, and B. E. Hirsch, "Area of and volume enclosed by digital and triangulated surfaces," in *Medical Imaging 2002: Visualization, Image-Guided Procedures, and Display, Proceedings of SPIE*, vol. 4681, 2002.

[7] J. C. Mullikin and P. W. Verbeek, "Surface area estimation of digitized planes," *Bioimaging*, vol. 1, no. 1, pp. 6–16, 1993.

[8] G. Windreich, N. Kiryati, and G. Lohmann, "Voxel-based surface area estimation: from theory to practice," *Pattern Recognition*, vol. 36, no. 11, pp. 2531–2541, 2003.

[9] J. Lindblad, "Surface area estimation of digitized 3D objects using weighted local configurations," *Image and Vision Computing*, vol. 23, no. 2, pp. 111–122, 2005.

[10] J. Ziegel and M. Kiderlen, "Estimation of surface area and surface area measure of three-dimensional sets from digitizations," *Image and Vision Computing*, vol. 28, no. 1, pp. 64–77, 2010.

[11] R. Klette and A. Rosenfeld, *Digital Geometry: Geometric Methods for Digital Picture Analysis.* San Francisco: Morgan Kaufmann, 2004.

[12] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 922–928.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 652–660.

[14] A. Rabbani, M. Babaei, R. Shams, Y. D. Wang, and T. Chung, "DeePore: a deep learning workflow for rapid and comprehensive characterization of porous materials," *Advances in Water Resources*, p. 103787, 2020.

[15] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[16] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," *Proceedings of SPIE, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006, 2019.

[17] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.