RECOVERING SUB-VOXEL GEOMETRY: LEARNED IMPLICIT QUADRICS FOR LARGE-SCALE 3D SCENES

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale 3D scene representations based on discrete voxelization inherently discard sub-voxel surface structure, limiting geometric fidelity and the accuracy of downstream physics-aware sensor simulation. We introduce QuadricUNet, a neural architecture that represents scene surfaces using compact, continuous second-order polynomial surfaces (quadrics) instead of discrete voxels. Given only a sparse semantic occupancy grid, QuadricUNet predicts the 10 coefficients of a quadric surface for each occupied voxel, without requiring dense point clouds, surface normals, or explicit meshes as input. Because each surface is a low-degree implicit polynomial, geometric operations act directly on the predicted coefficients. We develop a vectorized Newton-Raphson projection for dense surface sampling that is more than 11× faster than Marching Cubes sampling the same field, and an analytical ray-quadric intersection for LiDAR simulation that eliminates the need for mesh extraction. On large-scale outdoor scenes, QuadricUNet achieves a Chamfer distance of 4.91 cm and 90.25% F1@10 cm on SemanticKITTI, and 4.93 cm and 93.37% F1@10 cm on the Waymo Open Dataset, improving on Marching Cubes and NKSR when these baselines operate on the same voxel-only input. For LiDAR simulation, the analytical representation achieves a 0.0814 m depth MAE, and an SPVCNN segmentation model trained on its simulated sweeps reaches 0.48 macro-mIoU on real sweeps, compared with 0.37 and 0.33 for sweeps simulated from Marching Cubes and NKSR meshes (0.64 when trained on real sweeps). These results show that continuous quadric-based geometry supports both higher-fidelity reconstruction and sensor simulation from sparse scene representations.

1 INTRODUCTION

Discrete volumetric grids remain an important representation for large-scale 3D scenes. Semantic scene completion (Xia et al., 2023b; Li et al., 2023b), occupancy prediction (Tian et al., 2023; Wang et al., 2023), and generative scene models (Bian et al., 2025; Ren et al., 2024) all produce voxel grids as their primary output, because a regular grid gives deep networks a tractable, convolution-friendly structure over unbounded outdoor extents. However, dense grids scale cubically with resolution, so large scenes are modelled at coarse voxel sizes. This quantization discards the continuous surface geometry inside each voxel leading to the loss of sub-voxel boundaries, surface orientation, and thin structures. For tasks that only need occupancy, this loss is an acceptable tradeoff but for physics-based applications such as LiDAR simulation, it is not. Rays cast against voxelized geometry return staircase depth profiles and aliased occlusion boundaries (Whitaker, 2000; Lempitsky, 2010). Perception models trained on such data therefore face a sim-to-real gap that originates in the representation itself.

Recovering continuous surfaces from a voxel grid is a difficult problem. Existing approaches are best categorized by their input-output modalities. Methods that map a grid directly to an explicit surface, such as classical iso-surface extraction, operate at the resolution of the grid and reproduce its quantization (Lorensen & Cline, 1987; Whitaker, 2000). Methods that map point clouds to a continuous implicit field require dense points, often with accurate, oriented normals (Williams et al., 2022; Huang et al., 2023a). A semantic voxel grid supplies neither: points and normals estimated from voxel centres inherit the quantization of the grid, and the reconstruction propagates it. Learned

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

decoders conditioned on voxel grids or local features avoid this requirement, but they return a field that must be evaluated by a network at every query point (Peng et al., 2020; Chibane et al., 2020; Li et al., 2023a). Generative LiDAR models, finally, output point sets or range images directly, without an underlying surface (Caccia et al., 2019; Ran et al., 2024). None of these routes takes a semantic voxel grid as input and returns a surface on which geometric operations can be performed in closed form.

We propose to represent the surface inside each occupied voxel by a single implicit quadric, the zero set of a second-order polynomial with ten coefficients. Quadrics are closed under affine changes of coordinates and can represent many of the primitives found in outdoor environments, while degree two remains low enough for stable fitting and a compact representation (Wu et al., 2024; Xia et al., 2023a). Because each surface is an implicit polynomial with known coefficients, sampling reduces to projecting points onto a polynomial zero set, and ray intersection reduces to solving a scalar quadratic equation. This bypasses the need for intermediate meshing operations or network evaluation at query time. Our QuadricUNet predicts these coefficients from a sparse semantic occupancy grid alone. A single voxel is geometrically ambiguous in isolation: a locally flat patch may belong to a large plane, or the boundary of a thin object. We therefore regress coefficients with a 3D U-Net (Çiçek et al., 2016) whose receptive field supplies the regional context that a local prediction lacks.

We evaluate the representation along three axes: intrinsic reconstruction, LiDAR simulation, and computational cost. On SemanticKITTI and the Waymo Open Dataset, predicted quadrics recover sub-voxel geometry more accurately than baselines operating on the same voxel-only input. For simulation, segmentation models trained on sweeps ray-cast analytically against predicted quadrics transfer to real sweeps better than models trained on sweeps simulated from Marching Cubes or NKSR meshes. Operating directly on the polynomial also changes the cost of geometric operations. We find that projection-based sampling is more than an order of magnitude faster than Marching Cubes applied to the same field.

The primary contributions are as follows:

- A local implicit polynomial representation. One quadric per occupied voxel captures sub-voxel geometry with ten parameters and supports analytical sampling and ray intersection without mesh extraction or query-time network evaluation.
- A learning formulation for predicting quadrics from semantic voxels. It combines regional context from a 3D U-Net with normalized targets in coefficient space so predictions require neither dense points nor surface normals.
- An empirical study of reconstruction and sensor simulation from sparse scene representations. It shows that the quadric geometry improves both intrinsic surface recovery and sim-to-real transfer of LiDAR segmentation.

2 Related Work

Semantic voxel representations of scenes. Much of large-scale 3D scene understanding has relied on a common output format of semantic voxel grids. Semantic scene completion (Song et al., 2017; Xia et al., 2023b; Li et al., 2023b) infers a dense semantic grid from partial observations, occupancy prediction estimates it from camera or LiDAR input (Huang et al., 2023c; Tian et al., 2023; Wang et al., 2023), and recent generative models (Lee et al., 2024; Bian et al., 2025; Ren et al., 2024; Liu et al., 2024) synthesize entire driving scenes directly as semantic voxels. The format is convenient because it gives networks a regular structure over unbounded extents, but it answers only which voxels are occupied and by what. The geometry inside each voxel, where surfaces actually lie, is discarded. Recovering it is the problem this work addresses, and the natural first approach is to extract a surface from the grid itself.

Surfaces from volumetric grids. Iso-surface extraction methods such as Marching Cubes (Lorensen & Cline, 1987) and Dual Contouring (Ju et al., 2002) convert a sampled scalar field into a mesh, and learned variants (Chen et al., 2022; Shen et al., 2023) improve vertex placement and feature preservation within each cell. These methods, however, rely on a signal inside the cells, such as a distance value or a gradient, to place the surface between grid points. A binary occupancy grid

provides no such signal, so extraction reproduces the quantization of the grid rather than recovering the geometry beneath it. Continuous surfaces require a representation that is not tied to grid vertices, which is what implicit surface representations provide.

Implicit surfaces from points. Implicit representations describe a surface as the zero set of a continuous function. Classical reconstruction fits such a function to oriented points (Kazhdan & Hoppe, 2013), and neural methods learn it with networks: globally, with a single network per shape (Park et al., 2019; Mescheder et al., 2019); locally, with small decoders conditioned on features from a grid or a neighbourhood, which lets them scale to scenes (Peng et al., 2020; Chibane et al., 2020); or through learned kernels, which reach large outdoor environments (Williams et al., 2022; Huang et al., 2023a). This progression has steadily improved scalability, but two assumptions persist. Most methods require dense points, often with accurate oriented normals, which a semantic voxel grid cannot supply. Moreover, the surface is defined implicitly by network weights or a kernel expansion, so every geometric query must evaluate that function, and operations such as ray casting typically require an extracted mesh. An implicit surface that avoids both assumptions must be simple enough to be evaluated analytically, which leads back to an older line of work.

Algebraic and primitive-based surfaces. Before neural fields, implicit surfaces were often low-degree polynomials fitted to observed data. The literature on modelling implicit polynomials gives up the tools to fit them reliably, like distance approximations that correct the bias of algebraic residuals, and regularization that stabilizes higher-degree fits (Tasdizen et al., 2000; Taubin, 1991). Algebraic point set surfaces Guennebaud & Gross (2007) extended the idea locally, fitting spheres to point neighbourhoods, and primitive-based methods decompose shapes into planes, cylinders, superquadrics or convex parts, by robust fitting or with learned networks (Paschalidou et al., 2019; Chen et al., 2020; Sharma et al., 2020; Wu et al., 2024). Such surfaces are compact, and their geometry, including intersections with rays, is available analytically from a handful of coefficients. Their limitation is that they are fitted to, or detected in, observed geometry, so they presuppose the dense data that a voxel grid lacks. Our QuadricUNet combines the two directions: it keeps the analytic form of local quadrics but predicts one per voxel from semantic occupancy, using algebraic fitting only to produce training targets.

LiDAR simulation and sim-to-real transfer. The value of recovering analytic surfaces is clearest in sensor simulation, where fidelity is fundamentally geometric. Physics-based simulators cast rays against meshes of synthetic worlds or surfel maps reconstructed from real scans, so their realism is bounded by the geometry they intersect (Dosovitskiy et al., 2017; Manivasagam et al., 2020). Data-driven generators bypass geometry and synthesize range images or point clouds directly (Zyrianov et al., 2022; Ran et al., 2024), while neural fields render LiDAR by marching rays through a learned volume with repeated network queries (Huang et al., 2023b; Tonderski et al., 2024). When simulated sweeps differ from real ones, perception models trained on them degrade, which makes this sim-to-real gap a practical measure of geometric fidelity. Our approach sits on the physics-based side, but derives its geometry from a semantic voxel grid. Any scene expressed in this format can in principle be ray-cast in the same way, including the output of completion or generative models, with each intersection computed analytically rather than by marching or mesh traversal.

3 METHOD

Our goal is to recover the surface geometry inside each voxel of a semantic occupancy grid, in the form of quadrics. We introduce the local quadric representation (Section 3.1), describe how reference quadrics are fitted to aggregated LiDAR for supervision (Section 3.2) and how QuadricUNet predicts them from the voxel grid alone (Section 3.3), and derive the two operations used downstream: dense surface sampling and LiDAR ray intersection (Section 3.4).

3.1 LOCAL QUADRIC REPRESENTATION

Rather than describing the whole scene with a single continuous function, we attach an implicit surface to every occupied voxel. For a voxel v with centre $\boldsymbol{o}_v$ and edge length s, we express points in a local frame centred on the voxel, $\boldsymbol{p} = (x, y, z)^\top \in \mathcal{B} = [-s/2, s/2]^3$, and define the surface as

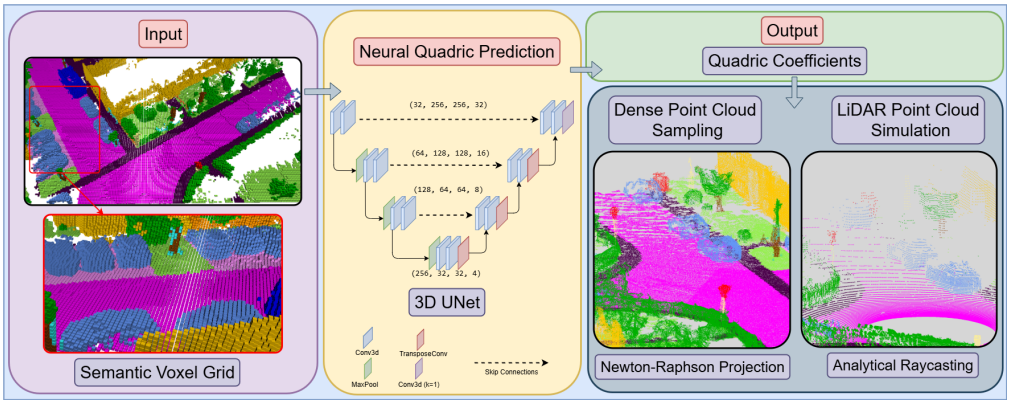

Figure 1: **Overview of QuadricUNet. Input:** a sparse semantic occupancy grid of $256 \times 256 \times 32$ voxels at 0.2 m, one-hot encoded by semantic class. The inset shows how voxelization reduces continuous surfaces to discrete steps. **Neural quadric prediction:** a 3D U-Net with three downsampling stages maps the grid to a coefficient field. **Output:** for every occupied voxel, the ten coefficients of a local quadric whose zero set $\Phi(\boldsymbol{p}; \boldsymbol{c}) = 0$ describes the surface within the voxel. During training, these are supervised by quadrics fitted to aggregated LiDAR. Because each surface is an implicit polynomial, two operators act on it directly, without mesh extraction: Newton-Raphson projection onto the zero set produces dense surface point clouds (*Dense Point Cloud Sampling*), and analytical ray-quadric intersection produces simulated LiDAR sweeps (*LiDAR Point Cloud Simulation*).

the zero set $\{\boldsymbol{p} \in \mathcal{B} : \Phi(\boldsymbol{p}; \boldsymbol{c}) = 0\}$ of a second-order polynomial:

$$\Phi(\boldsymbol{p}; \boldsymbol{c}) = Ax^2 + By^2 + Cz^2 + Dxy + Exz + Fyz + Gx + Hy + Iz + J = \boldsymbol{c}^\top \boldsymbol{m}(\boldsymbol{p}). \quad (1)$$

The coefficient vector $\boldsymbol{c} = (A, \ldots, J)^\top \in \mathbb{R}^{10}$ defines a quadric per voxel, where $\boldsymbol{m}(\boldsymbol{p}) = (x^2, y^2, z^2, xy, xz, yz, x, y, z, 1)^\top$. Two properties of this form are used throughout. First, Φ is linear in $\boldsymbol{c}$, which turns fitting into a least-squares problem. Second, Φ and its gradient $\nabla\Phi$, which gives the surface normal, are evaluated directly from the ten coefficients.

This representation sits between two lines of prior work. Implicit neural representations also describe surfaces as zero sets, but their functions are defined by network weights and must be evaluated by a forward pass at every query point. We keep an analytic form and make it local and learnable: a network predicts one polynomial per voxel. The choice of degree two polynomials balances expressiveness against stability. We discuss this choice in Section 4.4.

3.2 Fitting Quadrics to Aggregated LiDAR

To supervise the network, we fit a reference quadric to every occupied voxel using aggregated LiDAR point clouds, which are available only during training. Let $\{\boldsymbol{p}_i\}$ be the points supporting voxel v, expressed in its local frame. A point lies on the surface when its algebraic residual $\Phi(\boldsymbol{p}_i; \boldsymbol{c})$ is zero, so we seek the coefficients that make the residuals of all supporting points as small as possible. Minimizing the residuals alone, however, is not well posed. Since $\Phi(\boldsymbol{p}; \alpha\boldsymbol{c}) = \alpha\,\Phi(\boldsymbol{p}; \boldsymbol{c})$ for any $\alpha \neq 0$, all nonzero multiples of $\boldsymbol{c}$ share the same zero set, and the residuals can be made arbitrarily small by shrinking $\boldsymbol{c}$ towards the trivial solution $\boldsymbol{c} = \boldsymbol{0}$. We therefore restrict the coefficients to unit norm and minimize a weighted sum of squared residuals,

$$\boldsymbol{c}_v = \underset{\|\boldsymbol{c}\|_2 = 1}{\arg\min} \sum_i w_i\, \Phi(\boldsymbol{p}_i; \boldsymbol{c})^2, \quad (2)$$

where the weights w_i control the influence of each point and are described below. Since Φ is linear in $\boldsymbol{c}$, the objective in Eq. 2 is a quadratic form in $\boldsymbol{c}$, and its minimizer is the eigenvector with the smallest eigenvalue of the weighted scatter matrix $\sum_i w_i\, \boldsymbol{m}(\boldsymbol{p}_i)\boldsymbol{m}(\boldsymbol{p}_i)^\top$. In practice we add a small ridge penalty to this matrix for stability. The unit-norm constraint leaves only the sign of $\boldsymbol{c}_v$ undetermined, which we fix with an orientation convention (Appendix B).

The algebraic residual is not a geometric distance, so a few outlying points can pull the fit away from the surface. We therefore refine the weights by iteratively reweighted least squares, using Taubin's first-order approximation of the distance from a point to the zero set Taubin (1991),

$$d(\boldsymbol{p}; \boldsymbol{c}) = \frac{|\Phi(\boldsymbol{p}; \boldsymbol{c})|}{\|\nabla\Phi(\boldsymbol{p}; \boldsymbol{c})\|_2}. \tag{3}$$

Starting from weights that decrease with distance from the voxel centre, we alternate between solving Eq. 2 and down-weighting points with a large distance d to the current fit, so that outliers lose influence. Voxels with too few or nearly coplanar support points are assigned a plane, the special case of Eq. 1 with vanishing second-order terms. Scale and sign are fixed so that each reference quadric is unique (Appendix B).

3.3 Predicting Quadrics from Semantic Voxels

At inference time the only input is the semantic occupancy grid, encoding class and occupancy information. QuadricUNet maps this grid to ten coefficients at every voxel.

Predicting a quadric from a single voxel is ill-posed. A locally flat voxel may lie on a large plane, on a curve, or at the boundary of a thin object, and its own label cannot tell these cases apart. The information needed to resolve them is spread over neighbouring voxels, so we use a 3D U-Net whose encoder-decoder structure aggregates context over a large receptive field, with attention-gated skip connections that restore local detail (Appendix C, (Oktay et al., 2018; Çiçek et al., 2016)).

Because the ten coefficients differ widely in magnitude, the network predicts standardized coefficients $\hat{\boldsymbol{z}}_v$, which are compared with the standardized references $\boldsymbol{z}_v$ using a Huber loss ℓ_δ (Huber, 1964). A loss in coefficient space, however, does not weigh geometric errors uniformly. Planar surfaces dominate outdoor scenes, so most targets have vanishing second-order terms, and the network tends to under-predict curvature on the minority of curved surfaces. We therefore add a term that compares how strongly the predicted and reference quadrics bend, measured by the magnitude of their second-order coefficients, $h(\boldsymbol{c}) = \sqrt{4(A^2 + B^2 + C^2) + 2(D^2 + E^2 + F^2)}$, which is the Frobenius norm of the (constant) Hessian of Φ:

$$\mathcal{L} = \frac{1}{|\Omega|} \sum_{v \in \Omega} \Big[\bar{\ell}_\delta\big(\hat{\boldsymbol{z}}_v, \boldsymbol{z}_v\big) + \lambda\, \omega_v\, \ell_{\mathrm{sm}}\big(h(\hat{\boldsymbol{c}}_v), h(\boldsymbol{c}_v)\big) \Big]. \tag{4}$$

Here $\bar{\ell}_\delta$ averages the Huber loss over the ten coefficients, $\hat{\boldsymbol{c}}_v$ denotes the prediction mapped back to the original scale, and ℓ_{sm} is the smooth-L_1 loss (Girshick, 2015). Because most targets are planar, with $h(\boldsymbol{c}_v) = 0$, an unweighted curvature term would act mainly as a penalty pushing predictions towards flatness. We therefore weight each voxel by the curvature of its target,

$$\omega_v = \min\big(1 + \kappa\, h(\boldsymbol{c}_v),\ \omega_{\max}\big), \tag{5}$$

which equals 1 for planar targets, grows linearly with curvature at rate κ, and saturates at $\omega_{\max}$, so that curved surfaces receive more emphasis without dominating training. The weight depends only on the target and is not differentiated. Finally, λ balances the two terms.

We use standard augmentations, such as random flips and 90° rotations about the vertical axis. However, since each quadric is expressed in its voxel's local frame, the reference coefficients must be transformed together with the grid, which for these operations amounts to sign changes and reorderings of coefficient entries (Appendix D). We also apply voxel dropout, removing a random fraction of occupied voxels from the input and from supervision, so that predictions remain robust when parts of the neighbourhood are missing.

3.4 Operating on Predicted Quadrics

We now derive the two operations built to operate on the predicted quadric coefficients: sampling dense surface points, and intersecting LiDAR rays with the surface. Both use only the polynomial of Eq. 1 and its gradient.

Surface sampling. Sampling the surface requires points that satisfy $\Phi(\boldsymbol{p}) = 0$. Linearizing Φ around a point, $\Phi(\boldsymbol{p} + \boldsymbol{\delta}) \approx \Phi(\boldsymbol{p}) + \nabla\Phi(\boldsymbol{p})^\top \boldsymbol{\delta}$, the smallest correction that zeroes the linear model

lies along the normal $\nabla\Phi(\boldsymbol{p})$, and its length is the Taubin distance of Eq. 3. Iterating this correction gives a damped Newton update,

$$\boldsymbol{p}^{(k+1)} = \boldsymbol{p}^{(k)} - \eta \, \frac{\Phi(\boldsymbol{p}^{(k)}; \boldsymbol{c})}{\|\nabla\Phi(\boldsymbol{p}^{(k)}; \boldsymbol{c})\|_2^2} \, \nabla\Phi(\boldsymbol{p}^{(k)}; \boldsymbol{c}), \tag{6}$$

with damping factor $\eta \in (0, 1]$. For a plane the linearization is exact and a single undamped step reaches the surface; for curved quadrics the iterates converge locally, and they are restricted to their voxel. Starting from candidates spread over each voxel, a few iterations yield a dense sampling of the surface, and since each update involves only Φ and $\nabla\Phi$, all candidates are updated in parallel.

Ray intersection. In the local frame of voxel v, a LiDAR ray with origin $\boldsymbol{o}$ and unit direction $\boldsymbol{d}$ is $\boldsymbol{p}(t) = \boldsymbol{p}_0 + t\boldsymbol{d}$, with $\boldsymbol{p}_0 = \boldsymbol{o} - \boldsymbol{o}_v$ and range $t > 0$. Because Φ has degree two, its Taylor expansion along the ray is exact:

$$\Phi\big(\boldsymbol{p}(t)\big) = q_2\, t^2 + q_1\, t + q_0, \qquad q_0 = \Phi(\boldsymbol{p}_0), \quad q_1 = \nabla\Phi(\boldsymbol{p}_0)^\top \boldsymbol{d}, \quad q_2 = \Phi_2(\boldsymbol{d}), \tag{7}$$

where Φ_2 denotes the second-order terms of Eq. 1. The intersections are the roots $t = \big(-q_1 \pm \sqrt{q_1^2 - 4q_2q_0}\big)/(2q_2)$. The ray misses the surface if the discriminant is negative, and when $q_2 = 0$ the equation becomes linear, with $t = -q_0/q_1$. We take the smallest positive root and accept it if the intersection lies inside the voxel. The simulated return of the ray is the nearest accepted hit over all voxels it crosses, labelled with that voxel's semantic class. Unlike implicit neural fields, which locate zero crossings by marching along the ray with repeated network queries, each ray-voxel pair here requires only three scalars, and all pairs are solved in one batched operation.

4 Experiments

We evaluate QuadricUNet on two questions: how accurately it recovers sub-voxel surfaces from a semantic voxel grid (Section 4.2), and whether LiDAR sweeps simulated from its predictions transfer to real data (Section 4.3). We then examine the modelling choices behind these results and the computational cost of the geometric operators (Section 4.4).

4.1 Evaluation Protocol

Datasets and preprocessing. We use SemanticKITTI (Behley et al., 2019), training on sequences 00-07, 09 and 10 and evaluating on sequence 08, and the Waymo Open Dataset (Sun et al., 2020) with its standard training and validation splits. The input to every method is the ground-truth semantic occupancy grid (SemanticKITTI's voxelized scene labels; voxelized aggregated labels on Waymo); no method observes LiDAR points, normals or meshes at inference time. For each frame, we build an aggregated point cloud by registering neighbouring sweeps, removing moving objects, homogenizing the density and denoising with an edge-preserving filter (Appendix A). This cloud supplies the reference quadrics for training and, restricted to the occupied voxels, the reference geometry for evaluation.

Baselines. No existing method predicts analytic surfaces from a semantic voxel grid alone; learned local decoders return fields that require query-time network evaluation (Section 2). We therefore compare against two reference points given the same occupied voxels. *Marching Cubes* extracts the iso-surface of the binary occupancy grid. *NKSR* reconstructs a surface from one point per occupied voxel, placed at the voxel centre, with estimated normals. For *NKSR* we use the pretrained 'kitchen sink' model, shown to generalize to outdoor environments. Both operate outside the regime for which they were designed and indicate what the voxel grid supports without a learned sub-voxel model. Surfaces are compared as point sets using Chamfer distance (Fan et al., 2017), HD95 (Maier-Hein et al., 2024) and F1@τ (Tatarchenko et al., 2019); definitions, averaging and sampling densities are given in Appendix F.

4.2 Sub-Voxel Surface Recovery

On both datasets, predicted quadrics reach a Chamfer distance of about 4.9 cm, roughly a quarter of the voxel size, and an F1 of about 90% at a 10 cm threshold (Table 1). Their Chamfer distance

Table 1: Sub-voxel reconstruction from a semantic voxel grid. All methods receive the same occupied voxels. Metrics are averaged over classes (macro average). Chamfer distance and HD95 in cm, F1 in %.

Dataset	Method	Chamfer ↓	HD95 ↓	F1@5cm ↑	F1@10cm ↑
SemanticKITTI	Marching Cubes	14.81	57.34	20.03	45.78
	NKSR	14.45	89.08	39.76	65.98
	QuadricUNet	**4.91**	**38.91**	**60.09**	**90.25**
Waymo Open	Marching Cubes	10.32	27.06	20.40	50.77
	NKSR	12.72	63.20	27.76	59.83
	QuadricUNet	**4.93**	**12.60**	**61.61**	**93.37**

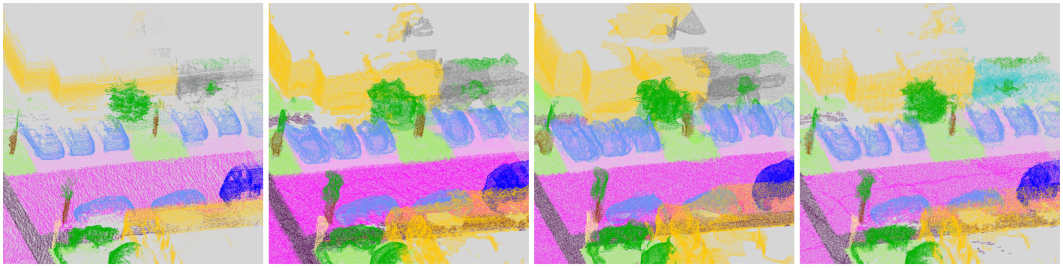

Figure 2: Qualitative reconstruction on SemanticKITTI. From left to right: aggregated LiDAR reference, Marching Cubes, NKSR and QuadricUNet. Marching Cubes reproduces the voxel grid as staircase surfaces, NKSR over-smooths across structures. Comparatively, QuadricUNet recovers vehicle contours and poles within each voxel with better accuracy.

is 52–67% lower than that of Marching Cubes, whose surfaces are tied to the voxel structure and therefore produce staircase artifacts (Figure 2). NKSR smooths across these steps but also across thin structures, because normals estimated from voxel centres carry the quantization of the grid into the reconstruction. The remaining error lies at finer scales (F1@5cm of about 60%). Since fitted quadrics reach 73.6% at this threshold (Table 4), this gap reflects prediction error rather than representational capacity.

4.3 LiDAR Simulation and Sim-to-Real Transfer

We simulate a 64-beam LiDAR modelled on the Velodyne HDL-64E, covering the full 360° field of view of the real scans (Appendix H). Rays are intersected analytically with predicted quadrics (Section 3.4) or cast against the extracted baseline meshes, and each return takes the semantic class of the voxel it hits.

Geometric fidelity. Against the real sweeps of sequence 08, analytic intersection with predicted quadrics gives the lowest depth error and the highest F1 at both thresholds (Table 2). Figure 3 shows why: meshes extracted from the voxel grid yield aliased, step-like returns, NKSR leaves holes where its surface is incomplete, and predicted quadrics reproduce continuous ground returns and object silhouettes.

Transfer to real data. For each of the four data sources, we train an SPVCNN (Tang et al., 2020) segmentation model with the same configuration on the SemanticKITTI training sequences and evaluate it on the real sweeps of sequence 08 (Table 3). The gain is broad: among the simulated sources, quadric-based sweeps give the highest IoU in 17 of the 19 classes, with the largest margins on trucks, motorcycles, fences, persons and vegetation (Appendix K). Thin and articulated structures remain difficult for every simulated source: bicycles reach 0.04 IoU against 0.32 with real data, Marching Cubes leads on bicyclists, and no model, including the one trained on real sweeps, recognizes motorcyclists.

Table 2: Geometric fidelity of simulated sweeps against the real sweeps of SemanticKITTI sequence 08: depth MAE (m) and point-level F1 (%) averaged over 19 classes. Predicted quadrics are best on all metrics.

Source	MAE ↓	mF1@10 ↑	mF1@20 ↑
Marching Cubes	0.1282	24.10	44.11
NKSR	0.0925	35.75	52.65
QuadricUNet	**0.0814**	**39.55**	**58.20**

Table 3: Sim-to-real transfer: SPVCNN trained on each source and evaluated on the real sweeps of sequence 08 (macro averaged). Our method recovers 42% of the gap between the strongest baseline and real data.

Training data	mIoU ↑	mAcc ↑
Marching Cubes	0.3672	0.4916
NKSR	0.3255	0.4532
QuadricUNet	**0.4806**	**0.5954**
Real (reference)	0.6352	0.7216

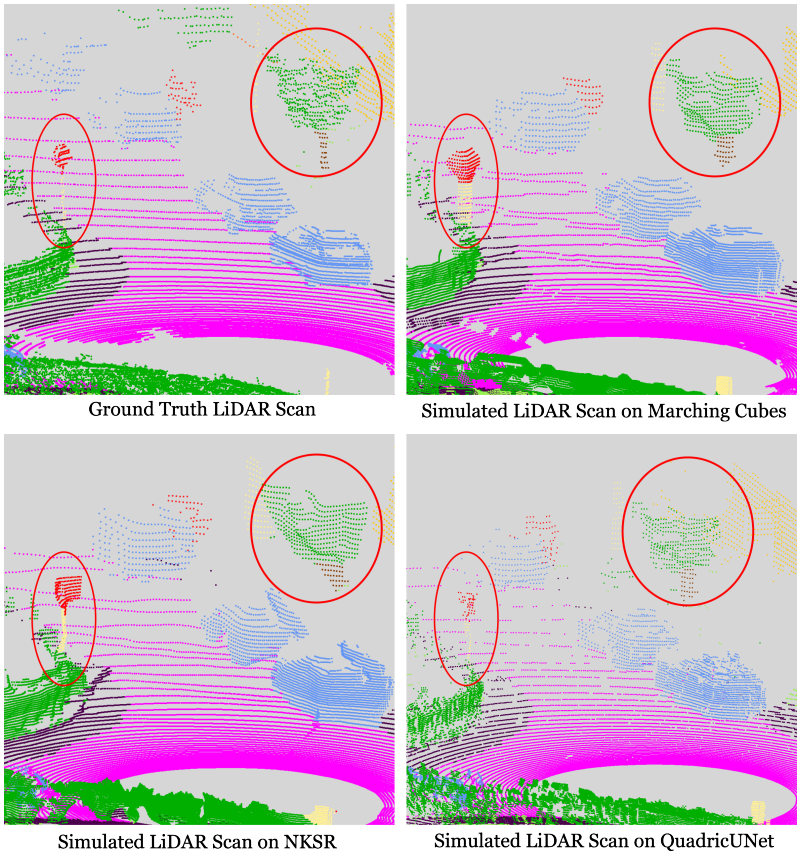

Figure 3: Simulated LiDAR sweeps on SemanticKITTI. Top: real sweep and Marching Cubes. Bottom: NKSR and QuadricUNet. Highlighted regions show the oversmoothing of structures present in outputs of NKSR and Marching Cubes which our method offsets with the ability to model the accurate sub-voxel surfaces.

4.4 Ablations and Computational Cost

Polynomial degree. Fitting polynomials of degree one to three directly to the aggregated points, without learning, separates representational capacity from learning (Table 4). Quadrics capture cylinders and curved patches that planes cannot represent, while cubics lose accuracy at the finest threshold, consistent with the known instability of higher-degree implicit polynomial fits (Tasdizen et al., 2000). Degree two thus offers the best balance of fine-scale fidelity, stability and compactness.

Table 4: Representational capacity by polynomial degree: the reference fitter (Appendix B) applied to all aggregated points without learning (Appendix J). Cubics double parameters and time and lose accuracy at 2 cm. Class averages; time to fit and sample 10^4 voxels.

Degree	Params	Time (s) ↓	Chamfer (cm) ↓	F1@2cm ↑	F1@5cm ↑	F1@10cm ↑
1 (plane)	4	1.13	5.08	23.01	65.25	87.44
2 (quadric)	10	2.30	4.34	**25.10**	**73.59**	92.50
3 (cubic)	20	4.93	**4.26**	21.27	73.44	**93.62**

Curvature-weighted objective. Removing the curvature term ($\lambda = 0$) degrades all three macro averages (Table 5). The per-class pattern shows that minority classes with higher curvatures benefit from the additional loss term, which follows the motivation of Section 3.3.

Table 5: Effect of the curvature term in Eq. (4) on SemanticKITTI (macro average; single runs). The curvature term improves all three metrics on 15 of the 19 classes (Table 8); road and sidewalk change by at most 0.03, and only the two rarest classes, bicyclist and motorcyclist, regress.

Objective	Chamfer (cm) ↓	F1@5cm (%) ↑	F1@10cm (%) ↑
Huber only ($\lambda = 0$)	4.95	59.48	89.99
Huber + curvature (ours)	**4.91**	**60.09**	**90.25**

Computational cost of vectorized projection. Newton projection samples the surface $11\times$ faster than iso-surfacing the same surface at equal fidelity via Marching Cubes, using more memory because all candidates are processed at once (Table 6).

Table 6: Dense surface sampling on the CPU from the same fitted reference quadrics, per frame. Newton projection evaluates only Φ and $\nabla\Phi$ at a fixed number of candidates per voxel, whereas Marching Cubes evaluates the field on a dense grid. Setup in Appendix I.

Method	Time (s) ↓	Memory (MB) ↓	Chamfer (cm) ↓	F1@10cm (%) ↑
Marching Cubes (same field)	77.21	**216**	**3.55**	94.57
Newton projection (ours)	**6.86**	1436	3.58	**94.68**

5 Limitations

A single quadric per voxel cannot represent three-plane corners, more than two surface sheets or detail finer than second order, which is consistent with the remaining errors on thin and articulated classes; adaptive resolution/octree style partitioning could allocate several quadrics to such regions. Continuity across voxel faces is also not enforced: neighbouring reference quadrics share support points, but the predictions are not constrained to agree at shared faces. Finally, our evaluation covers static scenes and ground-truth occupancy grids without a mechanism to evaluate the validity of occupied voxels.

6 Conclusion

QuadricUNet predicts one implicit quadric per occupied voxel of a semantic occupancy grid. From 20 cm voxels, it recovers surfaces to within about 5 cm, and LiDAR simulated from these surfaces transfers substantially better to real data than LiDAR simulated from baseline reconstructions of the same voxels. Natural next steps are dynamic scenes and grids produced by scene completion or generative models.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods, providing feedback on the evaluation methodology and experimental protocol and assisting in formulating and presenting the mathematical exposition of the method. We have not used generative AI tools for developing the theoretical models or conceptual framework, proposing or refining research hypotheses, or supporting qualitative and thematic data analysis. The use of generative AI tools for generating synthetic datasets, providing critical ingredients for or writing proofs, translation, and cleaning or reformatting datasets is not applicable to this work. Additionally, we used generative AI tools for creating and editing software code, drafting and editing the text of the paper, and identifying and summarizing relevant literature. We have reviewed all AI-assisted work. AI-generated code was verified and tested by the authors; all AI-drafted text and mathematical statements were checked by the authors against the implementation. We take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All experiments use the public SemanticKITTI and Waymo Open datasets with their standard splits (Section 4.1). The construction of the aggregated reference point clouds and of the network input, including the treatment of moving objects and the class mappings, is described in Appendix A. The fitting of reference quadrics, with all parameters, is given in Section 3.2 and Appendix B. The network architecture, objective and optimization settings are given in Section 3.3 and Appendix C, and the coefficient transformations used for augmentation are listed in Appendix D. The two geometric operators are derived in Section 3.4, with their implementation parameters in Appendix E. The evaluation protocol, metric definitions, sampling densities and baseline configurations are specified in Section 4.1 and Appendices F and G, and the LiDAR simulation, segmentation training and timing setups are given in Appendices H and I. All results are from single training runs. We will release the source code for data preparation, quadric fitting, training, evaluation and simulation, together with trained models, upon publication.

REFERENCES

Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Hengwei Bian, Lingdong Kong, Haozhe Xie, Liang Pan, Yu Qiao, and Ziwei Liu. DynamicCity: Large-scale 4d occupancy generation from dynamic scenes. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=M7KyLjuN0A.

Lucas Caccia, Herke van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of lidar data. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5034–5040. IEEE Press, 2019. doi: 10.1109/IROS40897.2019.8968535. URL https://doi.org/10.1109/IROS40897.2019.8968535.

Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Trans. Graph.*, 41(4), July 2022. ISSN 0730-0301. doi: 10.1145/3528223.3530108. URL https://doi.org/10.1145/3528223.3530108.

Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image*

Computing and Computer-Assisted Intervention – MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II, pp. 424–432, Berlin, Heidelberg, 2016. Springer-Verlag. ISBN 978-3-319-46722-1. doi: 10.1007/978-3-319-46723-8_49. URL https://doi.org/10.1007/978-3-319-46723-8_49.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3D object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Trans. Graph.*, 26(3): 23–es, July 2007. ISSN 0730-0301. doi: 10.1145/1276377.1276406. URL https://doi.org/10.1145/1276377.1276406.

Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4369–4379, 2023a. doi: 10.1109/CVPR52729.2023.00425.

Shengyu Huang, Zan Gojcic, Zian Wang, Francis Williams, Yoni Kasten, Sanja Fidler, Konrad Schindler, and Or Litany. Neural lidar fields for novel view synthesis. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18190–18200, 2023b. doi: 10.1109/ICCV51070.2023.01672.

Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9223–9232, 2023c. doi: 10.1109/CVPR52729.2023.00890.

Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL https://doi.org/10.1214/aoms/1177703732.

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Trans. Graph.*, 21(3):339–346, July 2002. ISSN 0730-0301. doi: 10.1145/566654.566586. URL https://doi.org/10.1145/566654.566586.

Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), July 2013. ISSN 0730-0301. doi: 10.1145/2487228.2487237. URL https://doi.org/10.1145/2487228.2487237.

Jumin Lee, Sebin Lee, Changho Jo, Woobin Im, Juhyeong Seon, and Sung-Eui Yoon. Semcity: Semantic scene generation with triplane diffusion. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28337–28347, 2024. doi: 10.1109/CVPR52733.2024.02677.

Victor Lempitsky. Surface extraction from binary volumes with higher-order smoothness. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1197–1204, 2010. doi: 10.1109/CVPR.2010.5539832.

Pengfei Li, Ruowen Zhao, Yongliang Shi, Hao Zhao, Jirui Yuan, Guyue Zhou, and Ya-Qin Zhang. Lode: Locally conditioned eikonal implicit scene completion from sparse lidar. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8269–8276. IEEE, 2023a.

Yiming Li, Zhiding Yu, Christopher Choy, Chaowei Xiao, Jose M. Alvarez, Sanja Fidler, Chen Feng, and Anima Anandkumar. VoxFormer: Sparse Voxel Transformer for Camera-Based 3D Semantic Scene Completion. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9087–9098, Los Alamitos, CA, USA, June 2023b. IEEE Computer Society. doi: 10.1109/CVPR52729.2023.00877. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.00877.

Yuheng Liu, Xinke Li, Xueting Li, Lu Qi, Chongshou Li, and Ming-Hsuan Yang. Pyramid diffusion for fine 3d large scene generation. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXIX*, pp. 71–87, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-72889-1. doi: 10.1007/978-3-031-72890-7_5. URL https://doi.org/10.1007/978-3-031-72890-7_5.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, August 1987. ISSN 0097-8930. doi: 10.1145/37402.37422. URL https://doi.org/10.1145/37402.37422.

Lena Maier-Hein, Annika Reinke, Patrick Godau, Minu D Tizabi, Florian Buettner, Evangelia Christodoulou, Ben Glocker, Fabian Isensee, Jens Kleesiek, Michal Kozubek, et al. Metrics reloaded: recommendations for image analysis validation. *Nature methods*, 21(2):195–212, 2024.

Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4455–4465, 2019. doi: 10.1109/CVPR.2019.00459.

Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174, 2019. doi: 10.1109/CVPR.2019.00025.

Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10336–10345, 2019. doi: 10.1109/CVPR.2019.01059.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III*, pp. 523–540, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58579-2. doi: 10.1007/978-3-030-58580-8_31. URL https://doi.org/10.1007/978-3-030-58580-8_31.

Haoxi Ran, Vitor Guizilini, and Yue Wang. Towards realistic scene generation with lidar diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14738–14748, June 2024.

Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. XCube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4209–4219, 2024. doi: 10.1109/CVPR52733.2024.00403.

Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. ParSeNet: A parametric surface fitting network for 3d point clouds. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII*, pp. 261–276, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58570-9. doi: 10.1007/978-3-030-58571-6_16. URL https://doi.org/10.1007/978-3-030-58571-6_16.

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Trans. Graph.*, 42(4), July 2023. ISSN 0730-0301. doi: 10.1145/3592430. URL https://doi.org/10.1145/3592430.

Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas Funkhouser. Semantic Scene Completion from a Single Depth Image . In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 190–198, Los Alamitos, CA, USA, July 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.28. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.28.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3D architectures with sparse point-voxel convolution. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, pp. 685–702, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58603-4. doi: 10.1007/978-3-030-58604-1_41. URL https://doi.org/10.1007/978-3-030-58604-1_41.

T. Tasdizen, J.-P. Tarel, and D.B. Cooper. Improving the stability of algebraic curves for applications. *IEEE Transactions on Image Processing*, 9(3):405–416, 2000. doi: 10.1109/83.826778.

Maxim Tatarchenko, Stephan R. Richter, Rene Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What Do Single-View 3D Reconstruction Networks Learn? . In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3400–3409, Los Alamitos, CA, USA, June 2019. IEEE Computer Society. doi: 10.1109/CVPR.2019.00352. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00352.

G. Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, 1991. doi: 10.1109/34.103273.

Xiaoyu Tian, Tao Jiang, Longfei Yun, Yucheng Mao, Huitong Yang, Yue Wang, Yilun Wang, and Hang Zhao. Occ3D: A large-scale 3d occupancy prediction benchmark for autonomous driving. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=ApqgcSnhjh.

Adam Tonderski, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. NeuRAD: Neural rendering for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14895–14904, June 2024.

Xiaofeng Wang, Zheng Zhu, Wenbo Xu, Yunpeng Zhang, Yi Wei, Xu Chi, Yun Ye, Dalong Du, Jiwen Lu, and Xingang Wang. OpenOccupancy: A large scale benchmark for surrounding semantic occupancy perception. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17804–17813, 2023. doi: 10.1109/ICCV51070.2023.01636.

Ross T. Whitaker. Reducing aliasing artifacts in iso-surfaces of binary volumes. In *Proceedings of the 2000 IEEE Symposium on Volume Visualization*, VVS '00, pp. 23–32, New York, NY, USA, 2000. Association for Computing Machinery. ISBN 1581133081. doi: 10.1145/353888.353893. URL https://doi.org/10.1145/353888.353893.

Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18500–18510, June 2022.

Ji Wu, Huai Yu, Wen Yang, and Gui-Song Xia. QuadricsNet: Learning concise representation for geometric primitives in point clouds. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4060–4066, 2024. doi: 10.1109/ICRA57147.2024.10610722.

Chao Xia, Chenfeng Xu, Patrick Rim, Mingyu Ding, Nanning Zheng, Kurt Keutzer, Masayoshi Tomizuka, and Wei Zhan. Quadric representations for lidar odometry, mapping and localization. *IEEE Robotics and Automation Letters*, 8(8):5023–5030, 2023a. doi: 10.1109/LRA.2023.3290510.

Zhaoyang Xia, Youquan Liu, Xin Li, Xinge Zhu, Yuexin Ma, Yikang Li, Yuenan Hou, and Yu Qiao. Scpnet: Semantic scene completion on point cloud. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17642–17651, 2023b. doi: 10.1109/CVPR52729.2023.01692.

Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII*, pp. 17–35, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20049-6. doi: 10.1007/978-3-031-20050-2_2. URL https://doi.org/10.1007/978-3-031-20050-2_2.

A Data and Preprocessing

Grid. Each sample covers $x \in [0, 51.2]$ m, $y \in [-25.6, 25.6]$ m and $z \in [-2.0, 4.4]$ m in the sensor frame, discretized into $256 \times 256 \times 32$ voxels of $s = 0.2$ m.

Aggregation on SemanticKITTI. For each frame, we aggregate preceding scans whose poses lie within 150 m of the current pose, retaining a scan whenever the vehicle has moved at least 1.5 m since the last retained one. Points closer than 2.5 m or farther than 80 m from the sensor are discarded, as are points labelled unlabeled, outlier or other-object. Points of moving classes are removed from all aggregated scans except the current one; the corresponding voxels are removed during input construction (below). All points are transformed into the sensor frame of the current scan and cropped to the grid.

Aggregation on Waymo. For each segment, all frames with 3D semantic labels (first return of the top LiDAR) are aggregated in the world frame. Moving objects are suppressed per instance: an instance is removed if its horizontal extent over the segment exceeds a class-specific threshold (6.5 m for cars; 2.5 m for other vehicles, motorcyclists and bicyclists; 0.8 m for trucks). With this threshold, nearly all truck instances are removed, and buses and pedestrians are not filtered. The aggregated cloud is then transformed into the vehicle frame of each labelled frame and cropped to the grid.

Density homogenization and denoising. Point density is homogenized by spatial hashing on a 5 cm grid, keeping one point per occupied cell. Sensor noise is reduced with a bilateral filter over the 20 nearest neighbours of each point. The spatial weight is a Gaussian of the neighbour distance ($\sigma_s = 5$ cm), and the range weight is a Gaussian ($\sigma_r = 5$ cm) of the neighbour's offset along the direction from the local centroid to the point.

Input construction. On SemanticKITTI, the input is derived from the dataset's voxelized scene labels. Labels are mapped to the 19 evaluation classes, a merged class for outlier, other-structure and other-object, and an empty class (21 input channels). Voxels labelled as moving are removed, as are non-ground voxels that contain no aggregated LiDAR point. On Waymo, the input is obtained by voxelizing the labelled aggregated cloud, with each voxel taking the label of a point it contains. Classes are one-hot encoded into 22 channels, which merges the sidewalk and walkable classes. On both datasets, the input and supervised voxel sets are restricted to voxels for which a reference quadric could be fitted (Appendix B).

B Reference Quadric Fitting

Support. On both datasets, the support of voxel v consists of all aggregated points within 0.3 m of its centre, so neighbouring voxels share points. Voxels with fewer than 5 support points receive no target.

Planar case. Let $\lambda_1 \le \lambda_2 \le \lambda_3$ be the eigenvalues of the covariance of the support points, with corresponding eigenvectors, and let $\nu_v = \lambda_1/(\lambda_1+\lambda_2+\lambda_3)$. If $\nu_v < 0.015$ or the support contains fewer than 20 points, the target is the plane through the centroid $\bar{\boldsymbol{p}}$ with normal $\boldsymbol{n}$ equal to the eigenvector of λ_1: $\boldsymbol{c}_v \propto (\mathbf{0}_6^\top, \boldsymbol{n}^\top, -\boldsymbol{n}^\top\bar{\boldsymbol{p}})^\top$. Besides stability, this rule addresses a degeneracy of Eq. (2): if the support points lie on a plane, every product of that plane with another linear function also passes through them, so the smallest eigenvalue is repeated and the minimizer is not unique.

Weighted fit. Eq.(2) is solved in normalized coordinates $\hat{\boldsymbol{p}} = (2/s)\,\boldsymbol{p}$, in which the voxel spans $[-1,1]^3$, with the ridge penalty

$$\boldsymbol{\Gamma} = \gamma\,\mathrm{diag}(10\cdot\mathbf{1}_6,\ \mathbf{1}_4), \qquad \gamma = 10^{-4},$$

added to the weighted scatter matrix. The penalty is ten times stronger on the second-order coefficients, which biases the fit towards flatter solutions unless the points require curvature. The solution $\hat{\boldsymbol{c}}$ is mapped back to metric local coordinates by $\boldsymbol{c}_v \propto \boldsymbol{S}\hat{\boldsymbol{c}}$ with $\boldsymbol{S} = \mathrm{diag}\big((2/s)^2\mathbf{1}_6,\ (2/s)\mathbf{1}_3,\ 1\big)$, and renormalized to unit norm.

Reweighting. The initial weights are $w_i^{(0)} = \exp(-\|\boldsymbol{p}_i\|_2^2/\sigma_s^2)$ with $\sigma_s = 0.2\,\mathrm{m}$. After each solve, they are updated as $w_i^{(k+1)} = w_i^{(0)}\exp\big(-d(\boldsymbol{p}_i;\boldsymbol{c}^{(k)})^2/\sigma_f^2\big)$ with $\sigma_f = 0.1\,\mathrm{m}$ and d the distance of Eq. (3). We run 10 iterations.

Orientation. Since $\boldsymbol{c}$ and $-\boldsymbol{c}$ define the same surface, we choose the sign for which the gradient at the voxel centre, $\nabla\Phi(\mathbf{0}) = (G,H,I)^\top$, points towards the sensor: $(G,H,I)\,(\boldsymbol{o}_\mathrm{s} - \boldsymbol{o}_v) \ge 0$. The same rule applies to planar targets. We take $\boldsymbol{o}_\mathrm{s}$ to be the origin of the sensor frame on SemanticKITTI and the origin of the vehicle frame on Waymo, where it approximates the LiDAR position. With scale and sign fixed, each target is unique whenever the smallest eigenvalue is simple; the planar case covers the degenerate configurations.

C Network and Training

Architecture. QuadricUNet is a dense 3D U-Net with three downsampling stages. Each stage uses a residual block of two $3\times3\times3$ convolutions with batch normalization and ReLU, a $1\times1\times1$ residual projection, and 3D dropout, followed by $2\times2\times2$ max pooling. Encoder widths are 32, 64 and 128 channels, with a 256-channel bottleneck at $32\times32\times4$. The decoder upsamples with transposed convolutions. At each level, the skip features pass through an additive attention gate conditioned on the upsampled decoder features, and are concatenated with them before a residual block. A $1\times1\times1$ convolution outputs the ten standardized coefficients at every voxel.

Objective. Coefficients are standardized per entry with the mean and standard deviation of the reference coefficients over the training set. In Eq. (4) we use Huber threshold $\delta = 0.1$ and $\lambda = 5$, and in Eq. (5) $\kappa = 10$ and $\omega_{\max} = 5$.

Optimization. We train with AdamW (weight decay 10^{-4}), a one-cycle learning-rate schedule peaking at 10^{-3} with cosine annealing, batch size 4, gradient clipping at 1.0 and fp16 mixed precision. Training used a single NVIDIA H200 GPU.

D Augmentation

Each of the following transformations is applied independently with probability 0.5: a reflection of the grid along x, a reflection along y, and a 90° rotation about the vertical axis. In voxel-local coordinates, these map $x \mapsto -x$, $y \mapsto -y$ and $(x,y) \mapsto (-y,x)$, respectively. Substituting into Eq. (1) gives the corresponding transformations of the coefficients:

$$\begin{aligned} x \mapsto -x:&\quad (D,E,G) \mapsto (-D,-E,-G),\\ y \mapsto -y:&\quad (D,F,H) \mapsto (-D,-F,-H),\\ (x,y) \mapsto (-y,x):&\quad (A,B,D,E,F,G,H) \mapsto (B,A,-D,-F,E,-H,G), \end{aligned}$$

with all other coefficients unchanged. Each map only changes signs and reorders entries, so it preserves the unit norm. Because the sensor position is transformed together with the scene, the orientation convention is preserved as well. Standardization is applied after augmentation. Voxel dropout is applied to a sample with probability 0.2 and removes between 5% and 7% of its voxels from both the input and the supervised set.

E Geometric Operators

Surface sampling. For each voxel, 800 candidates are drawn uniformly in a box of edge $1.2s$ centred on the voxel. We apply $K = 3$ updates of Eq. (6) with $\eta = 0.85$, and skip the update wherever $\|\nabla\Phi\|_2^2 \le 10^{-6}$. For a plane, each damped step reduces the distance to the surface by a factor of $1 - \eta$. Candidates that end outside the voxel are discarded, and 400 points per voxel are drawn with replacement from the remaining ones. Voxels whose predicted surface does not intersect the voxel contribute no points.

Ray intersection. Candidate ray-voxel pairs are found with axis-aligned tests against the exact voxel bounds. The equation in Eq. (7) is treated as linear when $|q_2| \le 10^{-7}$.

F Evaluation Protocol Details

Reference. For each frame, the reference point set $\mathcal{R}$ consists of the aggregated points (Appendix A) that lie inside occupied voxels with a semantic label.

Metrics. Let $d(\boldsymbol{x}, \mathcal{Y}) = \min_{\boldsymbol{y}\in\mathcal{Y}} \|\boldsymbol{x} - \boldsymbol{y}\|_2$. For a sampled point set $\mathcal{P}$, the Chamfer distance is

$$\mathrm{CD}(\mathcal{P}, \mathcal{R}) = \frac{1}{2}\Big(\frac{1}{|\mathcal{P}|}\sum_{\boldsymbol{x}\in\mathcal{P}} d(\boldsymbol{x}, \mathcal{R}) + \frac{1}{|\mathcal{R}|}\sum_{\boldsymbol{y}\in\mathcal{R}} d(\boldsymbol{y}, \mathcal{P})\Big).$$

HD95 is the larger of the 95th percentiles of $\{d(\boldsymbol{x}, \mathcal{R})\}_{\boldsymbol{x}\in\mathcal{P}}$ and $\{d(\boldsymbol{y}, \mathcal{P})\}_{\boldsymbol{y}\in\mathcal{R}}$. For a threshold τ, precision is the fraction of $\mathcal{P}$ within τ of $\mathcal{R}$, recall is the fraction of $\mathcal{R}$ within τ of $\mathcal{P}$, and F1@τ is their harmonic mean. Metrics are computed per frame, averaged over frames within each class, and then averaged over classes.

Sampling densities. QuadricUNet surfaces are sampled with 400 points per occupied voxel (Appendix E). Baseline meshes are sampled uniformly by area with 3×400 points per occupied voxel. The higher density does not disadvantage the baselines.

G Baselines

Marching Cubes. The occupied voxels are written into a zero-padded binary grid, and the isosurface at level 0.5 is extracted with scikit-image. Mesh vertices are mapped from grid indices to scene coordinates, and the same meshes are used for LiDAR simulation.

NKSR. We use the released NKSR reconstructor with its default pretrained weights, without retraining. The input is one point per occupied voxel, at its centre, with a normal estimated by Open3D principal component analysis (radius 0.6 m, at most 30 neighbours) and oriented towards the sensor-frame origin. Reconstruction uses a 0.2 m voxel size and dual-mesh extraction.

H LiDAR Simulation

Scan pattern. The simulated sensor has 64 beams: 32 with elevations spaced linearly from $+2.0^\circ$ to -8.83°, and 32 from -9.33° to -24.33°. Azimuth is sampled every 0.2°, giving 115,200 rays per sweep, all emitted from the sensor origin.

Scene extent. To cover the full 360° field of view of the real scans, each simulated scene merges two grids: the grid of the current frame, which covers the area ahead of the vehicle, and the grid of the earlier frame transformed into the current frame, which covers the area behind it. Each method is applied to both grids independently, and the two predictions (quadrics for QuadricUNet, meshes for the baselines) are placed in the current frame. For frames without such an earlier frame, only the forward grid is used.

Ray casting and labels. Baseline meshes are ray-cast with Open3D's `RaycastingScene`. Each return takes the semantic class of the voxel it hits; returns in voxels without a semantic label are discarded.

Fidelity metrics. Depth MAE is computed by projecting simulated and real sweeps onto a common spherical range image's elevation and azimuth bins and averaging the absolute range difference over bins that contain a return in both sweeps. Point-level mF1 is averaged over 19 classes.

Segmentation training. All four SPVCNN models use the same configuration as the original SPVCNN training. The training data are the SemanticKITTI training sequences, either simulated from each source or real. Evaluated on the real sweeps of sequence 08 over the 19 evaluation classes, with per-class scores averaged over instances (Table 7).

I Cost Measurement

Timings in Table 6 are averaged over 1000 random frames of SemanticKITTI, and exclude the network forward pass. Both methods run on the CPU, using NumPy for projection and scikit-image for Marching Cubes.

J Polynomial Degree Study

For Table 4, every degree is fitted with the procedure of Appendix B: the same support definition, planar rule, weighted fit with ridge penalty, iterative reweighting and orientation convention, with the monomial vector m(p) extended to degree d. Unlike the reference targets, no minimum support is imposed. Every voxel of the grid that contains aggregated points is fitted, whether or not it is occupied in the semantic grid, including voxels with fewer than 5 support points, which Appendix B leaves without a target. Voxels with fewer than 20 support points fall under the planar rule and receive a plane at every degree. Planes are sampled exactly. Quadrics and cubics are sampled by three undamped Newton steps from points on the tangent plane at the voxel centre, keeping points inside the voxel. Point labels are transferred from the labelled scan by nearest neighbour, and metrics are averaged over classes. Storage assumes 5 bytes of index and label data per voxel plus 4 bytes per coefficient. Times include fitting and sampling. Results are computed on 1000 randomly selected frames.Unlike the evaluations in Tables 1 and 6, which are restricted to occupied semantic voxels and exclude very sparsely supported ones, this study fits every voxel that contains aggregated points; its errors are therefore higher than those of the reference quadrics in Table 6.

K Per-Class Results

Table 7: Per-class sim-to-real transfer on SemanticKITTI. One SPVCNN model is trained on each data source (real sweeps, or sweeps simulated from Marching Cubes, NKSR or QuadricUNet), and all are evaluated on the real sweeps of sequence 08. Per-class mIoU and mAcc are means over all instances of the class across the scans; the macro average is taken over the 19 classes. Bold marks the best simulated source per class.

	Real sweeps		Marching Cubes		NKSR		QuadricUNet	
Class	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc
Car	0.9611	0.9918	0.9312	0.9729	0.9049	0.9492	**0.9483**	0.9803
Bicycle	0.3246	0.3732	0.0127	0.0158	0.0103	0.0143	**0.0420**	0.0533
Motorcycle	0.6368	0.8201	0.0338	0.0351	0.1572	0.3329	**0.3867**	0.5314
Truck	0.7188	0.9719	0.1559	0.4075	0.2048	0.5328	**0.5517**	0.9382
Other-vehicle	0.5643	0.6074	0.3441	0.4986	0.2643	0.3897	**0.3500**	0.3736
Person	0.7306	0.8368	0.2128	0.2291	0.0562	0.0710	**0.3830**	0.4190
Bicyclist	0.8942	0.9550	**0.5827**	0.7410	0.0000	0.0000	0.5233	0.5718
Motorcyclist	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Road	0.9347	0.9564	0.8161	0.9525	0.8232	0.9308	**0.8899**	0.9452
Parking	0.5003	0.5528	0.2654	0.3620	0.3218	0.3982	**0.4367**	0.5116
Sidewalk	0.8039	0.9155	0.5038	0.7863	0.6200	0.7991	**0.6802**	0.8839
Other-ground	0.0155	0.0247	0.0016	0.0075	0.0011	0.0018	**0.0030**	0.0049
Building	0.9056	0.9644	0.7871	0.8459	0.4862	0.5128	**0.8715**	0.9227
Fence	0.5991	0.7545	0.2474	0.6400	0.2557	0.6384	**0.4748**	0.7383
Vegetation	0.8800	0.9460	0.6704	0.7286	0.6696	0.8218	**0.8400**	0.9044
Trunk	0.6896	0.7891	0.3858	0.4420	0.3190	0.3810	**0.4652**	0.5316
Terrain	0.7523	0.8330	0.4579	0.5548	0.6047	0.7216	**0.6053**	0.7033
Pole	0.6318	0.7729	0.2870	0.7332	0.2571	0.7820	**0.3559**	0.8283
Traffic-sign	0.5250	0.6444	0.2815	0.3880	0.2289	0.3328	**0.3245**	0.4707
Macro average	0.6352	0.7216	0.3672	0.4916	0.3255	0.4532	**0.4806**	0.5954

Table 8: Per-class effect of the curvature term on SemanticKITTI reconstruction. Chamfer distance in cm, F1 in %. Bold marks the better value per class and metric.

	Huber only ($\lambda = 0$)			Huber + curvature		
Class	Chamfer ↓	F1@5 ↑	F1@10 ↑	Chamfer ↓	F1@5 ↑	F1@10 ↑
Car	4.76	62.47	92.38	**4.74**	**62.74**	**92.46**
Bicycle	5.54	49.73	87.28	**5.46**	**50.69**	**87.77**
Motorcycle	5.25	55.41	89.55	**5.17**	**56.94**	**89.85**
Truck	5.30	53.41	88.96	**5.16**	**55.61**	**89.60**
Other-vehicle	5.06	58.58	90.17	**4.98**	**59.72**	**90.63**
Person	5.14	55.98	90.58	**5.05**	**57.75**	**90.92**
Bicyclist	**6.54**	**32.97**	76.40	6.65	32.44	**77.25**
Motorcyclist	**5.03**	**60.11**	**91.07**	5.23	57.48	89.25
Road	2.92	**88.55**	98.62	2.92	88.53	**98.65**
Parking	3.78	75.18	95.05	**3.77**	**75.34**	**95.17**
Sidewalk	3.48	80.41	96.41	3.48	**80.43**	**96.45**
Other-ground	6.50	44.10	77.20	**6.26**	**44.90**	**78.48**
Building	4.68	62.38	92.99	**4.64**	**63.19**	**93.14**
Fence	4.98	58.74	90.80	**4.90**	**59.65**	**91.21**
Vegetation	5.65	49.84	87.71	**5.49**	**51.40**	**88.66**
Trunk	5.00	58.79	91.24	**4.96**	**59.61**	**91.32**
Terrain	4.35	66.58	93.04	**4.32**	**66.68**	**93.20**
Pole	4.95	60.28	90.62	**4.91**	**61.22**	**90.76**
Traffic-sign	5.17	56.62	89.70	**5.12**	**57.47**	**89.93**
Macro average	4.95	59.48	89.99	**4.91**	**60.09**	**90.25**