Task-Driven Implicit Representations for Automated Design of LiDAR Systems

Supplementary Material

A. Full Real-World Evaluation Details

In practice, our technique leverages simulations of task-specific scene classes to design LiDAR systems optimized for real-world deployments. In the main text, we evaluate our synthesized LiDAR systems on held-out test sets of objects, trajectories, and scenes; here, we demonstrate their effectiveness when implemented on physical hardware.

To do this, we evaluate the performance of our synthesized LiDAR systems designed for face scanning (Experiment A) using a real-world optical setup in Fig. 10. We emulate these designs with a single-photon avalanche diode module (Micro Photon Devices PDM series) paired with a pulsed laser emitter, providing timing resolution around 50ps. We configure this capture hardware in a confocal arrangement, which co-locates emitter and detector optics, using a two-axis galvanometer scanner (enabling $\pm 20^\circ$ deflection per axis $\approx 40^\circ$ total field of view). The galvo mirrors allow us to precisely steer the pulsed beam to learned sensor ray directions. We record photon-return histograms at each scan position queried by the respective face scanning system designs, and apply learned time gates to isolate the corresponding depth intervals.

We apply our capture configuration to a synthetic head model and visualize results in Fig. 10. For each sensor configuration (2, 4, and 10 learned sensors), we sample 576 total rays and compare against a uniform scanning baseline typical of mobile devices. We make three key observations. First, our learned designs consistently improve coverage, increasing the proportion of rays that intersect the face surface. Second, this concentrated sampling enables finer capture of key facial details, specifically around the nose, eyes, and mouth. Third, learned time gating makes our reconstructions robust to depth outliers from ambient light or sensor noise – for example, a depth discontinuity on the right side of the baseline scan is absent in our results, as improved time gating avoids capturing such noise for scenes within the training distribution. The system's 50 ps timing resolution does not significantly affect performance, and the benefits of our method persist in this real-world setting.

A.1. Real-World LiDAR Effects.

The timing jitter of our SPAD detector was 50 ps (FWHM) and the quoted pulsewidth of our laser was $<$100 fs (FWHM). The overall instrument response function (IRF) of the system was measured to be 50 ps (FWHM). We capture with a bin size of 8 ps. The beam diameter at exit is 1.2 mm with circularity $> 90\%$.

B. Extended Quantitative Evaluations

B.1. Experiment A: Face Scanning for Mesh Reconstruction

Summary: We first evaluate our technique on smartphone *flash-LiDAR design* for face scanning. We sample 50 face meshes from the Basel Face Model [38] normalized to 40cm height, centered at $z = 0$, and oriented at azimuth π. From these meshes, we learn an implicit LiDAR density, and generate new face scanning designs under two constraints: i) sensor diversity, by fitting varying number of Gaussian sensors, and ii) spatial sampling, by importance-sampling rays from learned designs under a fixed ray budget. We simulate ray–mesh intersections on 50 test faces and reconstruct meshes from intersected points via Delaunay triangulation; we then compute Chamfer distance [2] against ground truth meshes for evaluation.

Additional Details: We present complete mesh reconstruction results (measured by Chamfer distance) from Expt. A in Tab. 1. Chamfer distance is a metric that quantifies the geometric similarity between two point sets by computing the average nearest-neighbor distance in both directions. To validate our approach, we compare reconstruction quality using our learned computational sensor configurations against the conventional uniform-sampling strategy employed by smartphone LiDARs. We consider designs with 2, 4, and 10 sensors; in each case, a sensor is represented as a learned parametric Gaussian distribution in our six-dimensional design space, defined by a learned origin, field of view, time gate, and sampling weight (with FoV and time gates taken from the learned Gaussian 95% intervals in the respective angular and time dimensions). Under a fixed total ray budget, we draw rays from each sensor proportional to its learned weight, then evenly sample points within that sensor's field of view. Thus, the number of sensors reflects a hardware-design constraint (more sensors afford greater design flexibility), while the total ray count corresponds to the available dot projections and pixels for mobile LiDAR emission and detection.

We report the mean ± 2 SEM Chamfer distance (with error over randomly sampled face meshes in our test set) for each sensor design in Tab. 1. We observe in Tab. 1 that our method enables consistently improved reconstruction quality across hardware-design constraints and varying ray budgets, from 25 rays (simulating low-cost commercial LiDAR resolutions) to 576 rays (simulating mobile LiDAR resolution). Notably, our method requires drastically lower

bandwidth for each such sensor and ray sampling configuration, as shown in Tab. 2. These results reveal a tradeoff between hardware flexibility (sensor count, ray samples) and both task performance and bandwidth. A key advantage of our technique is the ability to rapidly explore and balance these constraints to tailor LiDAR designs for diverse requirements.

Table 1. **Experiment A – Chamfer distance in millimeters (mean ± 2 SEM) by LiDAR system type for face mesh reconstruction.** We evaluate reconstruction quality using our method with varying sensor-count constraints against uniformly and randomly sampled LiDAR ray baselines. Each approach was tested over ray-sample budgets from $5 \times 5 = 25$ (low-cost commercial LiDAR resolution) up to $24 \times 24 = 576$ (mobile LiDAR resolution). Our method enables consistently improved reconstruction across sensor design diversity and ray sampling constraints.

Method↓ / Rays→	**25**	**36**	**49**	**64**	**81**	**100**	**121**
Baseline - Random	108.20±7.25	76.81±1.63	76.27±1.62	67.70±1.16	58.29±2.83	60.57±1.82	43.24±1.93
Baseline - Uniform	63.58±1.41	68.75±2.73	52.57±3.06	47.17±1.86	42.20±1.84	32.28±1.75	41.38±2.14
Ours - 2 Sensors	51.77±2.03	58.19±2.07	54.75±1.51	50.95±2.24	55.80±1.58	50.17±1.49	37.56±2.14
Ours - 4 Sensors	50.21±1.86	55.31±1.60	57.94±1.66	48.35±1.73	45.98±2.26	30.25±1.81	32.87±1.24
Ours - 10 Sensors	48.44±2.81	43.03±1.47	39.37±1.42	42.91±1.81	37.92±1.84	28.47±1.07	28.91±1.23
	144	**169**	**196**	**225**	**256**	**289**	**324**
Baseline - Random	44.81±1.95	49.66±1.55	39.07±1.85	35.54±1.53	37.20±1.37	32.94±1.27	37.16±1.85
Baseline - Uniform	39.77±1.58	37.43±2.15	36.49±2.09	33.20±2.05	31.38±1.39	29.65±1.37	30.39±1.93
Ours - 2 Sensors	34.58±1.92	34.77±1.95	32.96±1.40	36.91±1.25	33.30±1.56	34.25±1.21	34.01±1.93
Ours - 4 Sensors	37.75±1.58	30.59±1.23	30.71±1.39	25.72±1.26	24.99±1.24	26.82±1.48	23.15±1.01
Ours - 10 Sensors	27.76±1.19	27.30±1.05	25.16±1.25	25.38±1.16	24.43±1.29	23.15±1.25	22.75±1.25
	361	**400**	**441**	**484**	**529**	**576**	
Baseline - Random	29.43±1.25	28.34±1.20	28.97±1.41	29.74±0.82	29.09±1.07	27.50±1.44	
Baseline - Uniform	32.79±1.46	28.93±1.11	27.02±1.41	26.29±1.54	26.23±1.39	25.73±1.23	
Ours - 2 Sensors	27.50±1.44	28.33±1.60	26.04±1.34	27.08±1.09	26.28±1.38	25.99±1.35	
Ours - 4 Sensors	28.01±1.19	27.77±1.29	22.59±1.01	22.87±1.07	24.49±0.92	21.65±1.03	
Ours - 10 Sensors	22.37±1.13	21.33±0.96	21.24±0.89	21.03±0.90	20.33±0.87	19.84±0.87	

Table 2. **Experiment A – Required bandwidth for face-mesh reconstruction (Mbps) across sensor and ray configurations.** By leveraging optimized time gating, our computational LiDAR design approach substantially lowers data throughput requirements for face-mesh reconstruction across ray budgets (25–576 rays), while maintaining or improving Chamfer-distance accuracy.

Method/Rays	**25**	**36**	**49**	**64**	**81**	**100**	**121**	**144**	**169**	**196**
Baseline - Random	0.66	0.95	1.29	1.69	2.14	2.64	3.19	3.80	4.46	5.17
Baseline - Uniform	0.66	0.95	1.29	1.69	2.14	2.64	3.19	3.80	4.46	5.17
Ours - 2 Sensors	0.11	0.16	0.23	0.28	0.36	0.44	0.55	0.66	0.74	0.86
Ours - 4 Sensors	0.11	0.18	0.23	0.31	0.37	0.50	0.56	0.65	0.80	0.91
Ours - 10 Sensors	0.11	0.14	0.20	0.25	0.31	0.42	0.50	0.61	0.66	0.80
	225	**256**	**289**	**324**	**361**	**400**	**441**	**484**	**529**	**576**
Baseline - Random	5.94	6.76	7.63	8.55	9.53	10.56	11.64	12.78	13.97	15.21
Baseline - Uniform	5.94	6.76	7.63	8.55	9.53	10.56	11.64	12.78	13.97	15.21
Ours - 2 Sensors	0.98	1.14	1.22	1.42	1.58	1.75	1.97	2.13	2.37	2.53
Ours - 4 Sensors	1.03	1.18	1.37	1.54	1.68	1.81	2.10	2.30	2.47	2.66
Ours - 10 Sensors	0.88	0.99	1.20	1.31	1.47	1.60	1.77	1.97	2.15	2.35

B.2. Experiment B: End Effector Tracking for Trajectory Modeling

Summary: We evaluate our technique for *distributed LiDAR system* design for robotic end-effector tracking. We use MuJoCo [46] to simulate a 7-DOF KUKA IIWA performing a pick-and-place task over 40 randomized trials, solving inverse kinematics at approach, transit, and descent waypoints to simulate realistic, physically-grounded robotic arm motion. We constrain sensors to be ceiling-mounted at $z = 1$m with azimuth in range $[0, 2\pi]$ rad, elevation in range $[-\pi/2, -\pi/3]$ rad, and max ToF 2m. We fit sensor Gaussians with learnable origin covariances over our implicit density, simulating distributed sensor placements. We then importance-sample rays within learned sensor regions under a fixed budget, and simulate ray–end-effector intersections on a held-out test set.

Additional Details: We reconstruct trajectories from intersection points via spline interpolation and compare them to ground truth trajectories using Fréchet distance [1]. Fréchet distance measures the similarity between two curves by computing the minimum leash length required to continuously traverse both simultaneously. As shown in Tab. 3, we compare three learned LiDAR configurations (2, 4, and 8 sensors) against uniform and random sampling baselines over ray budgets from 300 to 1500. Each learned sensor represents a Gaussian distribution in our six-dimensional design space, modeling a distributed LiDAR setup through a learnable origin region (mean and covariance), a field of view, and a time gate. Rays are drawn in proportion to each sensor's learned weight evenly across each respective learned origin region and FoV.

Tab. 3 reports the Fréchet distance (mean ± 2 SEM, error over randomly sampled trajectories in test set) for each LiDAR system at each sensor count and ray budget. Our learned configurations consistently yield lower trajectory-reconstruction error than uniform and random baselines across all constraints, and we further find that our automatically generated designs require vastly lower bandwidth due to improved time gating over conventional designs for a given number of sampled rays (Tab. 4).

Table 3. **Experiment B – Frechet distance in meters (mean ± 2 SEM) by LiDAR system type for robot end effector trajectory reconstruction.** We evaluate trajectory reconstruction quality using our method with varying sensor-count constraints and compare it against conventional uniformly sampled LiDAR rays and randomly sampled LiDAR rays. Each approach was tested over ray-sample budgets from 300 to 1500 sampled rays total over the learned sensors. Our computationally generated LiDAR systems enable consistently improved tracking under various sensor design and ray sampling constraints.

Method↓ / Rays→	**300**	**400**	**500**	**600**	**700**	**800**	**900**
Baseline - Random	51.28±15.04	68.92±9.77	41.48±14.01	48.24±11.42	62.36±14.55	29.11±10.79	24.57±9.01
Baseline - Uniform	66.30±9.99	44.86±9.43	42.38±12.90	38.92±11.27	33.79±11.40	42.06±6.64	28.70±6.33
Ours - 2 Sensors	44.67±12.86	33.44±8.34	31.03±10.26	29.85±7.54	27.30±8.26	22.47±8.22	24.81±7.59
Ours - 4 Sensors	36.65±8.57	30.93±8.12	31.29±7.85	37.51±8.70	32.31±8.44	17.99±8.29	18.95±8.52
Ours - 8 Sensors	29.84±8.08	32.44±7.82	23.53±7.46	27.20±7.69	18.46±7.96	23.72±8.53	20.07±8.58
	1000	**1100**	**1200**	**1300**	**1400**	**1500**	
Baseline - Random	30.95±8.93	46.35±14.01	22.93±3.79	25.27±8.47	13.52±2.98	34.22±6.94	
Baseline - Uniform	20.47±5.59	16.41±7.77	23.86±4.78	15.52±3.88	14.12±2.01	15.95±3.31	
Ours - 2 Sensors	23.94±8.30	16.37±7.96	15.78±7.87	14.29±8.21	14.87±8.10	14.37±7.97	
Ours - 4 Sensors	21.52±8.09	17.24±8.25	15.42±8.41	16.01±8.06	14.91±8.34	14.26±8.25	
Ours - 8 Sensors	20.76±8.18	19.84±7.82	15.19±8.33	18.59±8.18	18.56±8.27	13.72±8.35	

B.3. Experiment C: Robotic Warehouse Scanning for Object Detection

Summary: We evaluate *motion-adaptive scanning LiDAR* design in a simulated warehouse inspection setting. We simulate 100 warehouse scenes with parametrized shelf

Table 4. **Experiment B – Required bandwidth for end-effector trajectory reconstruction (Mbps) across sensor and ray configurations.** By leveraging optimized time gating in our learned LiDAR designs, our approach greatly reduces data throughput requirements for robot end-effector trajectory reconstruction across ray budgets (300–1500 rays).

Method↓ / Rays→	300	400	500	600	700	800	900	1000	1100	1200	1300	1400	1500
Baseline - Random	8.04	10.72	13.40	16.08	18.76	21.44	24.12	26.80	29.48	32.16	34.84	37.52	40.20
Baseline - Uniform	8.04	10.72	13.40	16.08	18.76	21.44	24.12	26.80	29.48	32.16	34.84	37.52	40.20
Ours - 2 Sensors	5.19	6.92	8.65	10.38	12.10	13.83	15.56	17.29	19.02	20.75	22.48	24.21	25.94
Ours - 4 Sensors	4.01	5.35	6.69	8.03	9.37	10.71	12.05	13.39	14.72	16.06	17.40	18.74	20.08
Ours - 8 Sensors	3.51	4.67	5.83	7.00	8.18	9.34	10.51	11.68	12.84	14.00	15.17	16.33	17.50

heights and box sizes, then learn an implicit density over ground robot motion ($x \in [0, X]$), with fixed azimuth $-\pi/2$ (representing scanning LiDAR), elevation in range $[0, \pi/2]$, and max ToF 5m. We perform motion-adaptive sensing by truncating the density at each robot position x and fitting Gaussian-parameterized rays (each defined by a learned elevation and time gate) over the position-truncated density. We simulate ray-mesh intersections on a held-out test set, and count ray–box hits as detections. Fig. 1 shows an example of our motion-adaptive LiDAR design; conditioning LiDAR designs on robot location allows us to propose well-shaped sensing strategies for different scene regions. As Fig. 6 demonstrates, this spatial adaptation enables more robust detection while reducing bandwidth requirements by $\sim 10\times$ under a fixed ray budget.

Additional Details: We report complete box-miss-rate results (mean ± 2 SEM for error over sampled warehouse scenes in the test set) for our spatially-adaptive warehouse ground robot setting in Tab. 5. Box miss rate measures the percentage of ground-truth boxes that go undetected. To validate our Gaussian-based approach, we compare it against conventional uniform scanning across ray budgets of 1 to 20. Here, each learned Gaussian is treated as an individual LiDAR ray, modeled in our six-dimensional design space by a fixed origin and azimuth, plus learnable elevation angle and time-gating parameter. Rather than sampling weights, we enforce the ray budget by choosing how many Gaussian components (rays) to learn, making the total ray count the key constraint on sampling density per spinning LiDAR cycle. We find our method enables more robust box detection for very low ray counts compared to the conventional baseline techniques; however, we find our method has greater variance in performance as well, perhaps due to slight overfitting to the training scenes. In addition to maintained or improved detection accuracy, our computational design approach also requires significantly lower bandwidth due to improved time gating for sampled rays (Tab. 6).

B.4. Experiment D: Emitter Design

Additional Details: Our task-driven implicit density also enables automated synthesis of alternative, constraint-aware emitters for a fixed detector configuration. In applications such as face scanning, emitter placement or power limits may prevent ideal co-located, collimated illumination. In Fig. 9, we show that, for a detector replicating a standard mobile-phone LiDAR, our method can fit various emitter distributions over the 6D design space, yielding single and multiple-flood illumination patterns, and even bistatic emitter designs subject to user-specified hardware constraints. We evaluate these designs in a physically-accurate light-transport simulator [39] and show in Fig. 9 that these alternative designs have minimal impact on reconstruction quality using the fixed detector design.

Table 5. **Experiment C – Box miss rate percentage (mean ± 2 SEM) by LiDAR scanning type for warehouse box detection in moving ground robots.** We compare box-detection miss rates between conventional uniform scanning (akin to autonomous-vehicle sensors) and our spatially-adaptive sampling method, varying the per-location ray count from 1 to 20. At each spatial location, we sample the specified number of rays per sensor; our learned spatially-adaptive sensor consistently reduces miss rates by concentrating samples on region-specific areas of interest.

Method↓ / Rays→	1	2	3	4	5	6	7
Baseline - Random	89.44±0.78	73.06±1.76	62.50±1.90	65.00±1.50	40.28±2.83	51.11±3.54	35.00±2.58
Baseline - Uniform	100.00±0.00	100.00±0.00	77.78±0.00	68.89±1.42	64.72±1.35	79.72±3.64	31.11±3.83
Ours - Spatially Adaptive	68.89±1.81	88.06±2.69	40.28±3.44	26.39±3.05	25.56±2.78	22.78±2.24	27.78±3.38
	8	**9**	**10**	**11**	**12**	**13**	**14**
Baseline - Random	23.89±2.46	23.89±2.46	21.94±2.03	25.00±1.91	26.94±2.63	16.11±2.63	16.11±2.63
Baseline - Uniform	55.28±4.46	28.61±3.37	29.17±3.71	9.72±2.41	25.56±2.89	4.44±1.92	12.22±2.61
Ours - Spatially Adaptive	15.56±3.06	20.83±4.07	18.06±3.35	11.11±2.87	7.78±2.28	6.39±2.37	10.00±2.49
	15	**16**	**17**	**18**	**19**	**20**	
Baseline - Random	11.39±2.18	19.72±1.86	17.78±1.92	8.89±1.42	8.89±1.42	17.50±2.23	
Baseline - Uniform	8.61±1.86	8.06±2.25	1.39±1.18	10.00±2.36	0.83±0.94	4.72±1.93	
Ours - Spatially Adaptive	15.00±2.82	13.06±2.86	6.94±2.21	6.94±2.21	10.83±2.03	11.39±2.03	

Table 6. **Experiment C – Required bandwidth for box detection (Mbps) across sensor and ray configurations.** By enabling improved sensor time gating, our computational LiDAR design approach greatly reduces data throughput requirements for warehouse box detection across ray budgets (1–20 rays per cycle), while maintaining or improving detection accuracy.

Method↓ / Rays→	1	2	3	4	5	6	7	8	9	10
Baseline - Random	0.32	0.64	0.96	1.28	1.60	1.92	2.24	2.56	2.88	3.20
Baseline - Uniform	0.32	0.64	0.96	1.28	1.60	1.92	2.24	2.56	2.88	3.20
Ours - Spatially Adaptive	0.12	0.16	0.24	0.26	0.28	0.29	0.33	0.36	0.39	0.42
	11	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Baseline - Random	3.52	3.84	4.16	4.48	4.80	5.12	5.44	5.76	6.08	6.40
Baseline - Uniform	3.52	3.84	4.16	4.48	4.80	5.12	5.44	5.76	6.08	6.40
Ours - Spatially Adaptive	0.43	0.45	0.47	0.49	0.49	0.54	0.55	0.56	0.59	0.60

B.5. Experiment E: Ray-Visibility Term Ablation

Additional Details: A key benefit of our technique is that it models both scene variability and occlusion factors, which are essential for robust LiDAR design. To demonstrate this, we compare our proposed density evaluation to an ablated formulation that removes the ray visibility term for a fixed-trajectory end-effector tracking setting in Fig. 11. We then sample distributed LiDAR configurations as rays from each implicit density and compare trajectory reconstruction using Fréchet distance across varying ray budgets.

We report full quantitative results for this ablation in Tab. 7. The ray visibility term enables physically grounded occlusion modeling and distinguishes our approach from scene-space methods that cannot represent origin-specific

LiDAR occlusion. To evaluate the effect of occlusion modeling, we learn two implicit densities: one using the full metric and one using the ablated version. For both, we consider the robotic end-effector tracking setting with a fixed trajectory. We assess density quality by sampling rays that represent single-pixel distributed LiDAR sensors in Fig. 11. The goal is to evaluate how well rays sampled from each density support occlusion-aware LiDAR-based tracking.

Table 7. **Experiment E – End-effector trajectory reconstruction error (Fréchet distance mean ± 2 SEM) for sampled distributed systems with and without ray visibility term.** We learn an implicit density for a single-trajectory robot end-effector tracking setting, and sample a different number of single-pixel LiDAR rays from each implicit density. The learned implicit density with our ray visibility term produces highly robust designs even for low numbers of sampled rays, demonstrating the importance of the ray visibility term in enabling occlusion-free LiDAR sensing.

Method↓ / Rays→	**60**	**70**	**80**	**90**	**100**	**110**	**120**
Ours - No Ray Visibility Term	0.38±0.11	0.26±0.08	0.17±0.05	0.14±0.04	0.12±0.04	0.09±0.02	0.09±0.02
Ours - With Ray Visibility	0.10±0.02	0.10±0.01	0.09±0.01	0.08±0.01	0.08±0.01	0.07±0.01	0.07±0.01
	130	**140**	**150**	**160**	**170**	**180**	**190**
Ours - No Ray Visibility Term	0.09±0.02	0.08±0.01	0.07±0.01	0.07±0.01	0.07±0.01	0.06±0.00	0.07±0.00
Ours - With Ray Visibility	0.07±0.01	0.07±0.01	0.07±0.01	0.07±0.01	0.07±0.01	0.07±0.01	0.07±0.01

We present the results from this ablation study in Tab. 7. We note that Fréchet distance mean and error is shown for a fixed trajectory scene over *sampled distributed LiDAR systems* (rather than over variable trajectories). We find that our learned implicit density with the ray visibility term enables consistently robust trajectory reconstruction even with low numbers of sampled single-pixel LiDAR rays. Of note, our sampled distributed LiDAR systems can eventually catch up to the occlusion-aware system, but require nearly $2\times$ more rays, as these occlusion-naive sampled rays are more prone to obstructed LiDAR visibility preventing accurate tracking.

C. Comparison Against Occupancy Grid Modeling

In this work, we model the full task of LiDAR *system design* within a *6D design space*. This distinguishes us from prior works [25, 26], which consider optimizing only *placement* of a fixed LiDAR within 3D *scene space*. By considering the complete problem of LiDAR system design, our work encompasses not only LiDAR placement, but also multi-sensor selection, scanning patterns, FoV, sampling frequency, time gating, and motion (among other dimensions).

A key benefit of our technique is that we model LiDAR design in a six-dimensional space of ray origin, angle, and time-of-flight: $(\mathbf{x}, \mathbf{a}, \tau)^\top = (x, y, z, \phi, \psi, \tau)^\top \in \mathcal{D}$. This distinguishes our method from existing baseline techniques, which primarily focus on scanning LiDAR placement optimization *in 3D scene space* for autonomous vehicles [25, 26]. Notably, these techniques optimize LiDAR placements in 3D scene space by learning a probabilistic occupancy grid (usually around an AV) and selecting LiDAR positions that resolve the most semantic uncertainty. However, by modeling only scene-space occupancy, these approaches cannot capture *view-dependent occlusion effects*, a core component of LiDAR system design. By capturing view-dependent LiDAR configurations in 6D space, we support design of distributed, in-motion, spatially-adaptive, multi-sensor, and bi-static systems that occupancy-grid techniques cannot adequately handle.

To illustrate this, we implement occupancy-grid baselines [25, 26] for our Experiment E setting (distributed LiDAR system design for robotic end-effector tracking). Since these baseline techniques offer only *placement optimization* and do not offer a method for LiDAR design, we extend the technique for system design by first learning a probabilistic occupancy grid over end-effector positions, and then importance-sample single-pixel LiDAR rays, i.e. sampling rays within the robot workspace that intersect high-probability voxels. We present statistics for sampled rays from each method in Tab. 8, and for downstream trajectory reconstruction accuracy from sampled LiDAR systems (comprised of varying numbers of sampled rays) in Tab. 9.

Table 8. **Sampled ray statistics from our method vs. occupancy grid-based LiDAR ray sampling for end effector tracking.** Occupancy-grid baselines are highly dependent on voxel size, and cannot model view-dependent effects; in contrast, our method can efficiently represent the continuous design space for LiDARs over 6D space, which allows for efficient and robust modeling of physically-grounded, occlusion-aware LiDAR designs.

	Surface hits with no occlusion modeling (%)	**Average Ray Visibility**	**Surface hits with occlusion modeling (%)**
Occupancy Grid (voxel size=0.2)	17.21 ± 1.18	0.42 ± 0.02	4.32 ± 0.64
Occupancy Grid (voxel size=0.15)	24.85 ± 1.35	0.40 ± 0.02	5.96 ± 0.74
Occupancy Grid (voxel size=0.1)	44.82 ± 1.55	0.37 ± 0.02	10.96 ± 0.98
Occupancy Grid (voxel size=0.05)	69.56 ± 1.44	0.39 ± 0.01	16.55 ± 1.16
Ours - Ablated Ray Visibility	72.63 ± 1.39	0.35 ± 0.01	13.48 ± 1.07
Ours	67.80 ± 1.46	0.68 ± 0.01	33.33 ± 1.47

Table 9. **Baseline comparison against occupancy grid-based distributed LiDAR system sampling for end effector trajectory reconstruction (Fréchet distance mean ± 2 SEM).** Our technique enables consistently improved trajectory reconstruction by efficiently representing the continuous, view-dependent and occlusion-aware space of LiDAR designs. Error is shown over *sampled LiDAR systems* for each respective method.

Method↓ / Rays Sampled →	**60**	**70**	**80**	**90**	**100**
Occupancy Grid (voxel size=0.2)	5.783 ± 6.221	4.714 ± 4.199	12.402 ± 21.283	1.173 ± 0.577	5.580 ± 9.667
Occupancy Grid (voxel size=0.15)	17.984 ± 29.420	1.274 ± 0.780	6.067 ± 10.318	0.727 ± 0.258	0.844 ± 0.339
Occupancy Grid (voxel size=0.1)	0.499 ± 0.248	0.311 ± 0.127	0.320 ± 0.212	0.276 ± 0.079	0.189 ± 0.115
Occupancy Grid (voxel size=0.05)	0.401 ± 0.395	0.287 ± 0.211	0.352 ± 0.182	0.249 ± 0.196	0.110 ± 0.031
Ours - Ablated Ray Visibility	0.382 ± 0.115	0.259 ± 0.082	0.171 ± 0.051	0.145 ± 0.043	0.120 ± 0.040
Ours	0.105 ± 0.015	0.096 ± 0.013	0.087 ± 0.010	0.083 ± 0.010	0.077 ± 0.008
	110	**120**	**130**	**140**	**150**
Occupancy Grid (voxel size=0.2)	1.473 ± 1.052	7.062 ± 12.530	1.181 ± 0.674	0.864 ± 0.619	8.967 ± 11.811
Occupancy Grid (voxel size=0.15)	0.786 ± 0.567	0.538 ± 0.287	4.705 ± 6.625	0.430 ± 0.129	0.306 ± 0.111
Occupancy Grid (voxel size=0.1)	0.142 ± 0.034	0.163 ± 0.051	0.125 ± 0.024	0.105 ± 0.019	0.099 ± 0.020
Occupancy Grid (voxel size=0.05)	0.121 ± 0.054	0.113 ± 0.036	0.084 ± 0.012	0.072 ± 0.012	0.078 ± 0.013
Ours - Ablated Ray Visibility	0.092 ± 0.021	0.089 ± 0.021	0.086 ± 0.022	0.078 ± 0.010	0.073 ± 0.005
Ours	0.074 ± 0.008	0.072 ± 0.006	0.073 ± 0.006	0.071 ± 0.006	0.073 ± 0.006

We make two key observations demonstrating the strength of our technique over prior occupancy grid-based works. First, occupancy-grid sampling is highly sensitive to voxel size. We observe in Tab. 8 that sampled rays from coarser voxel grids yield consistently lower surface hit rates; as a result, reconstructed trajectories from LiDARs sampled from these coarse grids yield consistently higher error (Tab. 9). This reliance on voxel size can lead to significant tradeoffs in representation size vs. accuracy in large and complex scenes, which our technique avoids by modeling LiDAR designs as *implicit representations*. Second, occupancy grid-based selection *cannot model view-dependent occlusion effects*. We observe in Tab. 8 that the average ray visibility of occupancy-grid baselines is consistently low *regardless* of voxel size, and is comparable to our technique **without the ray visibility term** (i.e. without modeling view-dependent occlusions). In contrast, our full technique enables consistently improved ray visibility, which then enables significantly more robust downstream trajectory reconstruction as shown in Tab. 9. Thus, by modeling LiDAR systems as *implicit densities* over a *6D space*, we enable more efficient, physically-grounded, and robust system design compared to prior methods.

D. Additional Baselines and Computational Analysis

We compare our method against several additional baselines: (1) end-to-end differentiable sensor optimization (E2E-DSO): gradient-based optimization of ray origins/angles through a differentiable 2D LiDAR model, (2) RL (PPO) actor-critic policy for sequential ray selection, and (3) CEM evolutionary search (ES). We evaluate on a 2D surface reconstruction task (scene shown in Fig. 12a) where each method selects 7 rays with $x \in [-1, 1]$ and $\theta \in (0, \pi)$ to minimize reconstruction error (2D surface reconstruction MSE from Gaussian kernel regression over ray-surface intersections). We evaluate over 10 sequential trials; in each trial, we impose a new constraint prohibiting LiDAR origins (x) in a random segment (red region in Fig. 12a).

The key advantage of our method is that it can efficiently search, store, and query the *full* LiDAR design space for a task (Sec. 5). While baselines achieve comparable performance for each given scene configuration (Fig. 12b), they require many additional ray-simulation queries to adapt when constraints change (Fig. 12c). Because we can model and query the full LiDAR design space, our method maintains stable accuracy under changing scene constraints *without additional simulation queries* (Fig. 12c).

Tab. 10 reports cumulative computational cost to reach MSE ≤ 0.025 under sequentially changing placement constraints. All methods rely on simulation sampling during training. However, we amortize this cost by first efficiently searching and representing the full LiDAR design space via normalizing flows; this enables rapid EM-based design proposal at test time with minimal additional compute. In contrast, constraint changes require baselines to fine-tune on new simulation ray samples, adding significant compute overhead or requiring potentially costly additional simulation queries.

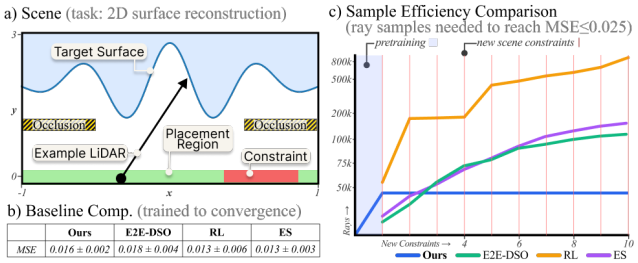

Figure 12. **Additional baseline comparisons.** (a) 2D surface reconstruction task with placement constraint (red). (b) All methods achieve comparable accuracy when trained to convergence. (c) Our method requires no additional simulation queries when constraints change, while baselines scale with the number of constraint changes.

Table 10. **Cumulative compute cost** to reach MSE ≤ 0.025 for 2D surface reconstruction under sequentially changing placement constraints. Our method's ray-sampling cost is fixed at pretraining; subsequent constraints require only lightweight EM fitting.

	Ours	**E2E-DSO**	**RL**	**ES**
Cumulative cost: Rays sampled (thousands) / GPU-hours / Wall-clock seconds				
N.Flow Pretraining	44.0/0.0024/9.14	*n/a*	*n/a*	*n/a*
New Constraint #1	44.0/0.0031/11.58	14.0/0.0021/7.59	55.1/0.0109/39.17	19.6/0.0011/3.86
New Constraint #5	44.0/0.0058/21.34	78.4/0.0163/58.63	426.0/0.0839/302.13	80.5/0.0043/15.36
New Constraint #10	44.0/0.0092/33.59	113.4/0.0214/76.97	890.6/0.1765/635.31	156.8/0.0083/29.88

E. Additional Qualitative Results

Experiment A. We provide additional qualitative results for our computationally designed sensors for face scanning in Fig. 13. Specifically, we visualize the LiDAR systems learned for face scanning across different sensor-count constraints. In this setting, each sensor is modeled as a learned parametric Gaussian in our six-dimensional design space (capturing its origin, field of view (FoV), and time gate); different sensor counts therefore represent hardware-design constraints and the feasible diversity of sensors in the face scanning system design. For each configuration, we render 196 sampled rays; while a higher ray budget would produce denser sampling within each sensor FoV, it would not affect the learned sensor layout itself.

We also present mesh reconstructions for each sensor configuration from the main paper over ray budgets ranging from $11 \times 11 = 121$ to $24 \times 24 = 576$ (the resolution typical of modern mobile LiDAR) in Fig. 14. Reconstructions are generated via Delaunay triangulation of the LiDAR–mesh intersection points. As the constraints on sensor count and ray budget are relaxed, our learned designs achieve increasingly detailed coverage of facial ge-

Figure 13. **Visualized sensor designs for face scanning.** We visualize sensor designs proposed by our technique. Each design is subject to a sensor count constraint, which represents a hardware-design constraint where more sensors afford greater design flexibility; each design is shown for the additional constraint of 196 total sampled rays.

ometry, particularly around the nose and eyes, by allocating dedicated sensors to these regions. Although more sophisticated reconstruction methods that incorporate priors can yield smoother meshes, our approach focuses on maximizing raw scene coverage; we therefore expect our sampling strategies to enhance performance irrespective of the downstream reconstruction algorithm.

Experiment B. We present supplementary visualizations of our computationally designed LiDAR sensors for robotic end-effector tracking in Fig. 15, across configurations with 2, 3, 4, 5, 8, and 10 learned distributed sensor regions. The learned designs adaptively distribute sensing capacity in response to the variance of pick-and-place trajectories. Under tight sensor-count constraints, sensors concentrate on low-variance segments (i.e., near the pick location and during mid-transfer) optimally balancing spatial coverage and time gating to minimize bandwidth. As constraints on sensor count relax, additional sensors are allocated to high-variance segments near the place location, improving tracking precision along the descent path.

Experiment C. We present a visualization of our learned spatially-adaptive sensor designs in Fig. 16. As shown in the main paper, our designs adapt to the heterogeneous spatial layout of the warehouse environment, conforming to both shelf heights and the expected sizes of boxes. To highlight the benefits of adaptive sampling, we visualize our ap-

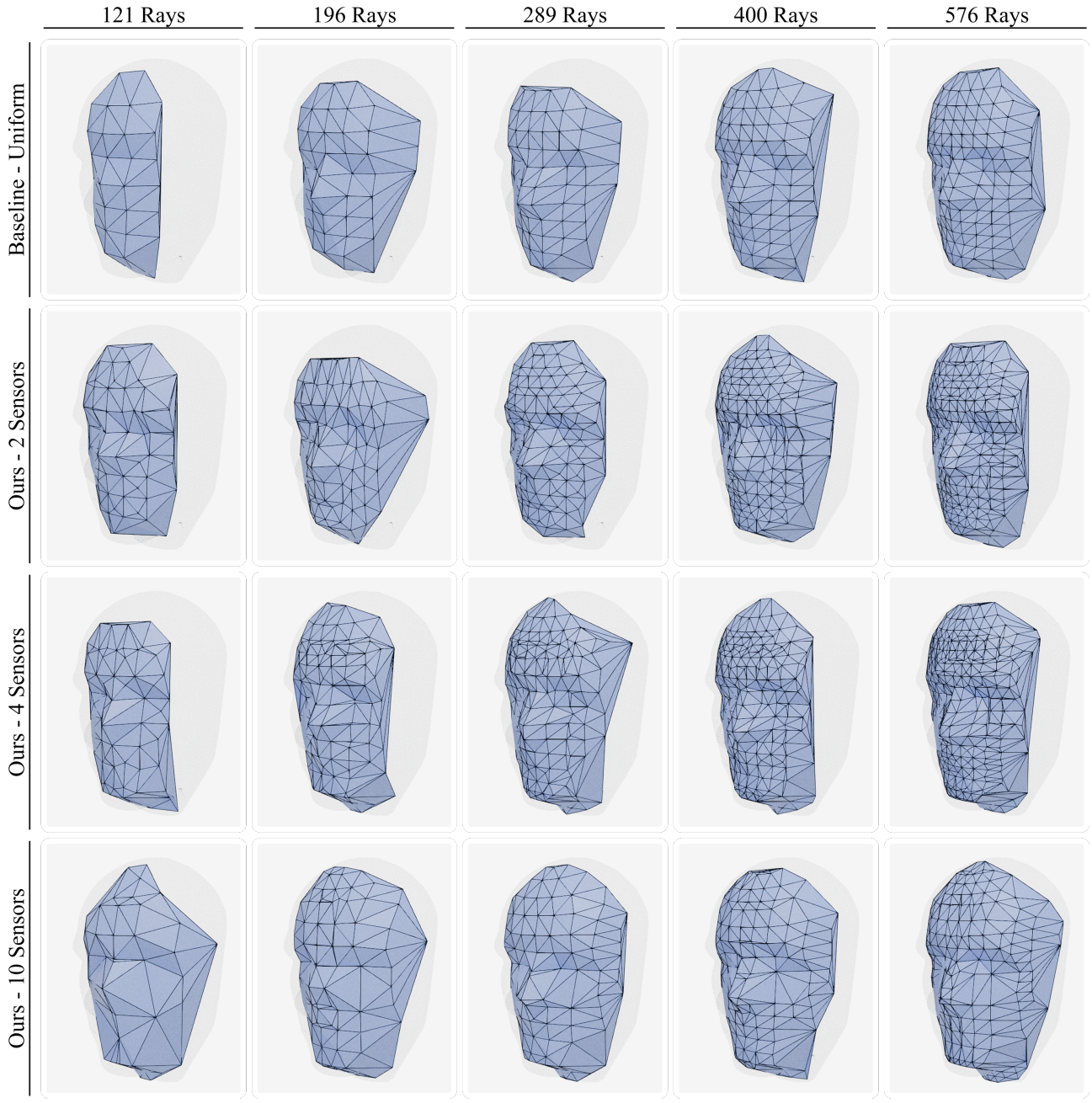

Figure 14. **Mesh reconstructions by sensor design and ray samples.** We visualize sample mesh reconstructions generated via Delaunay triangulation of LiDAR–mesh intersection points for a representative test face under various sensor configurations: the baseline uniform sampling (mimicking current mobile LiDARs) and our learned designs with 2, 4, and 10 sensors. As we relax constraints on sensor count and ray budget, our method optimally redistributes sampling to achieve more comprehensive coverage of the facial geometry, especially in peripheral face regions. Results for 196 rays are shown in the main paper.

proach against a baseline uniform scanning configuration using 10 and 30 rays per spatial point (with rays distributed across elevation angles at a fixed azimuth). Our spatially-adaptive designs concentrate samples on regions containing boxes, enabling superior box coverage and detection performance with fewer rays. Furthermore, intelligent time gating restricts sampling to time intervals most likely to intersect shelf and box surfaces, drastically reducing bandwidth requirements (as shown in Fig. 6). We thus demonstrate that our method can represent spatially-adaptive designs, which can achieve better object detection outcomes that minimize both LiDAR ray samples required and data throughput.

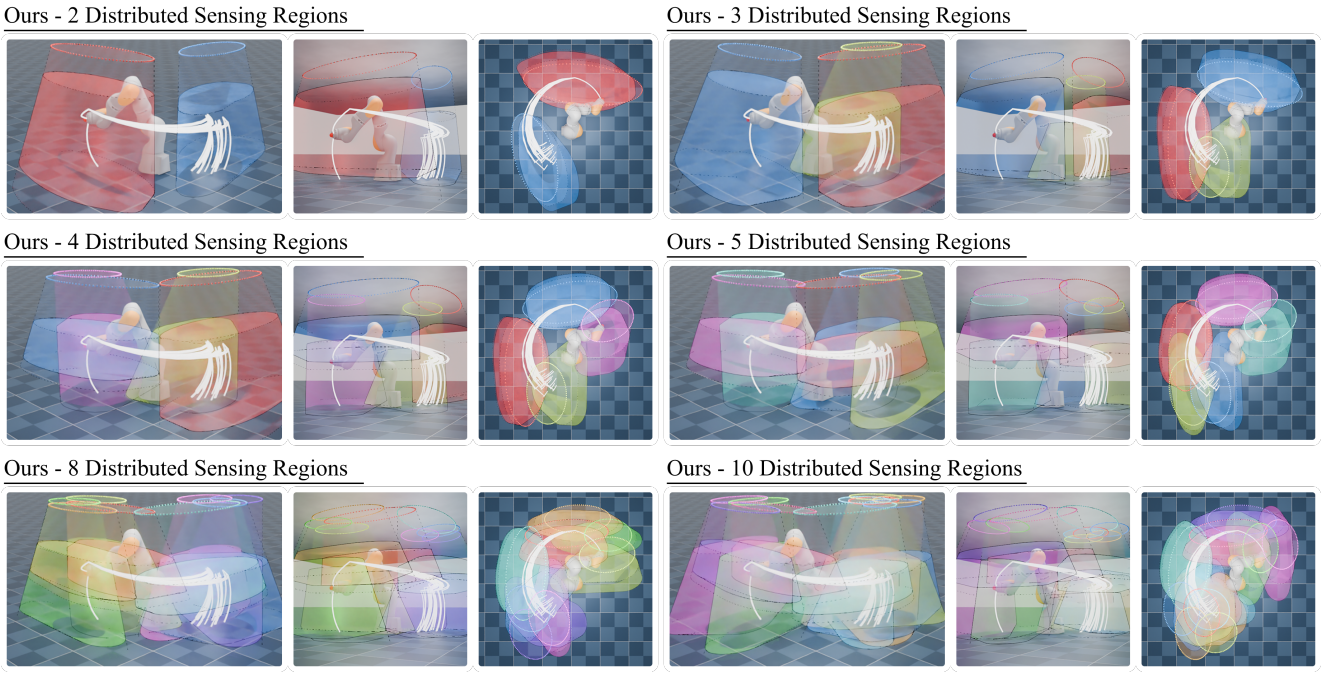

Figure 15. **Visualizations of learned LiDAR sensor regions for pick-and-place trajectories under 2, 3, 4, 5, 8, and 10 sensor constraints.** Learned distributed sensing systems show concentrated sensing on low-variance pick and mid-transfer segments and additional allocation to high-variance placement segments as constraints relax.

F. Additional Experimental Details

F.1. Experimental Settings

Face Scanning. We sample random shape coefficients from the Basel Face Model's [38] statistical PCA basis to reconstruct each head mesh; we then convert each mesh into a volumetric SDF which can be queried directly in our downstream implicit density learning.

Robotic Pick & Place. We specify a pick-and-place motion via six Cartesian waypoints: approach above the pick point, descent to grasp, lift, mid-transfer, pre-release descent, and final retreat. We solve for joint-angle configurations at each waypoint using a gradient-descent inverse-kinematics solver with MuJoCo's [46] forward dynamics and Jacobian computations, with damping and joint-limit clamping. We interpolate linearly in joint space between solutions to produce a smooth trajectory, and replay it in MuJoCo to obtain end-effector world-frame poses at every timestep, defining the LiDAR sampling path. In parallel, we build a serial kinematic chain from the robot URDF and wrap it into a volumetric SDF; for each joint vector along the trajectory, we freeze the chain's angles to instantiate a time-indexed sequence of signed-distance functions mapping any world-space point to its distance from the robot surface at that instant.

Warehouse Inspection. We simulate a warehouse setting with a linear array of shelves to emulate a ground robot equipped with scanning LiDAR navigating an aisle. Under a predefined parameter set of vertical spacings and box-height intervals, we uniformly sample shelves and boxes within the specified ranges, yielding spatially varying shelf geometry where specific regions have some range of shelf height, width, and box size characteristics. The assembled mesh is then converted into a volumetric SDF.

G. Toward Outdoor and Autonomous Vehicle Applications

Our framework can support semantic feature modeling in scenes beyond the indoor tasks evaluated in the main paper. To demonstrate this, we visualize the design-space density for LiDARs in the real-world nuScenes dataset in Fig. 17. Since pre-captured datasets use fixed LiDAR poses (preventing full design-space search as noted in Sec. 6), we instead visualize a simplified 3D design space (angle+time) for each fixed sensor origin $\mathbf{x}$. We learn a class-conditioned flow density from the dataset samples that reveals class-dependent structure in LiDAR design space, potentially informing unique sensor design for safety-critical actors (e.g., pedestrians). Our 6D implicit representation has a physical interpretation in 3D scene space, which could potentially support extensions modeling phenomena common in large-scale outdoor scenes, such as noise and multipath returns; we leave these directions to future work.

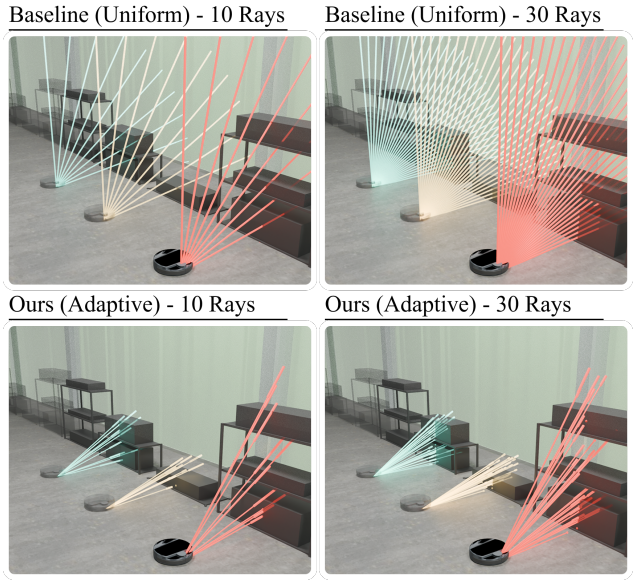

Figure 16. **Visual comparison of baseline uniform and our spatially-adaptive LiDAR** scanning configurations in a warehouse setting, with 10 and 30 rays per spatial point (sampled across elevation angles at fixed azimuth).

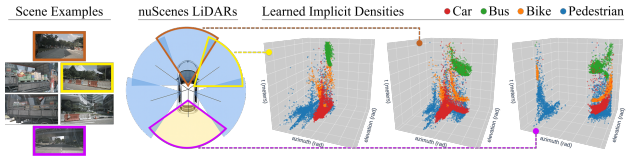

Figure 17. **Class-conditioned implicit LiDAR densities learned from real-world nuScenes data.** We visualize learned densities for four object classes (car, bus, bike, pedestrian), revealing class-dependent structure in LiDAR design space that could inform safety-critical sensor design.

G.1. Architecture and Training

Model Architecture. We define a six-dimensional normalizing flow over the dimensions $(x, y, z, \phi, \psi, \tau)$ of the LiDAR design space. A uniform base distribution is placed on each free coordinate (those whose configured lower and upper bounds differ), with any fixed dimensions (i.e., dimensions specified as fixed under user constraints) handled as constants. The flow architecture is composed of eight successive circular autoregressive rational-quadratic spline layers: each layer contains two autoregressive blocks whose conditioning networks have 64 hidden units and ReLU activations, and each block parameterizes a piecewise-rational-quadratic monotone spline with 16 bins. Azimuth and elevation are treated as circular variables, so the spline enforces periodic continuity on those axes. We use a tail-bound equal to the half-range of each active dimension, a 0.05 dropout rate, and learnable binary-mask permutations between layers to improve expressivity. We apply learnable shifts before and after each spline block to center the input and output spaces; this preserves invertibility, allows us to compute the Jacobian determinant analytically, and thus enables exact density evaluation and straightforward sampling of our task-driven implicit density.

Training. We train the flow for 5000 gradient steps using Adam with a 0.001 learning rate and a batch size of 1024. We add an entropy regularization term ($\eta_{\text{ent}} = 0.5$) to the negative log-likelihood to encourage exploration of the latent space. Ray visibility contributions to the target density are held at zero for the first 1000 iterations to concentrate learning on geometry; the visibility term is then linearly ramped to the full weight of 1.0 over 2000 steps. We use an SDF-based target with a Gaussian observation model ($\sigma = 0.025$) and incorporate visibility via transmittance computed by 32 stratified ray samples per point, each weighted by a sigmoid occupancy with $\beta = 100$. We ran each experiment on a single NVIDIA GeForce GTX 1080 Ti GPU (12 GiB VRAM).

H. Impact and Limitations

Our technique enables automated, task-specific LiDAR design tailored to diverse applications. By synthesizing sensor configurations that respect physical, cost, and form-factor constraints, it can reduce development time and material waste while supporting assistive robotics, environmental monitoring, and safer autonomous navigation. Constraint-aware design may also allow LiDAR deployment in previously inaccessible settings, such as small edge devices, low-power drones, or crowded indoor environments, broadening the reach of 3D sensing for social good. However, generating adaptive LiDAR configurations carries ethical responsibilities. Applications such as face scanning demand strict privacy, consent, and data-security protocols to prevent designs that could facilitate leakage of sensitive biometric patterns. Furthermore, projecting scene geometry into implicit sensor distributions risks embedding biases from simulated datasets, underscoring the need for rigorous dataset curation to ensure equitable performance across diverse populations and environments.

Limitations: Our framework relies on a simplified LiDAR physics model that omits effects such as angle and material-dependent reflectance (BRDF/retroreflectivity) and volumetric scattering; accurately modeling these phenomena is a valuable direction for future work. Additionally, our task-driven densities are learned from simulated scene distributions (represented with SDFs) which may not reflect real-world variability. While large-scale real-world validation remains an important next step, evaluation on existing pre-captured datasets (e.g., KITTI [10]) is not directly feasible, since our flow-based model requires actively querying the continuous 6D LiDAR design space. Extending our technique to large-scale real-world deployments is an important direction for future research.