SuperDec: 3D Scene Decomposition with Superquadric Primitives

Supplementary Material

1. Superquadrics

While our architecture can be easily adapted to segment and predict in an unsupervised manner other types of geometric primitives - in SuperDec we decided to use superquadrics. When looking for a suitable geometric primitive for our approach we were keeping in mind two main criteria. First, we wanted the primitive to be represented by a compact parameterization so that it can be described by only using a few parameters. Second, we wanted the representation to be expressive, in order to be able to describe real-world objects by only using a few primitives. Inspired by 3DGS [18], the first parameterization we took into consideration were the *ellipsoids*. Ellipsoids have a very compact parameterization as their shape can be represented using the following implicit equation:

$$f(\mathbf{x}) = \left(\frac{x}{s_x}\right)^2 + \left(\frac{y}{s_y}\right)^2 + \left(\frac{z}{s_z}\right)^2 = 1,$$

where the only free variables are s_x, s_y, s_z, which are the lengths of the three main semi-axis. However, if we start thinking about which objects and object parts can be effectively fitted using a single ellipsoid, we realize that their representational capabilities are not enough. In order to obtain higher representational capabilities while still keeping a simple representation, a natural extension are *generalized ellipsoids*. In this representation, we not only allow the length of the semi-axis to be variable, but their roundness controlled by the three exponents, which previously were fixed to 2. In that way, we obtain the following implicit function:

$$f(\mathbf{x}) = \left(\frac{|x|}{s_x}\right)^{e_1} + \left(\frac{|y|}{s_y}\right)^{e_2} + \left(\frac{|z|}{s_z}\right)^{e_3} = 1\,.$$

Using generalized ellipsoids with high exponents it becomes possible to also represent cuboidal shapes. While having suitable representational capabilities, these primitives do not allow to compute distance to their surface in a closed form, a property which can be extremely useful for various downstream applications. This drawback is overcome by *superquadrics*, at the cost of one less degree of freedom, which however does not substantially impact expressivity. Unlike generalized ellipsoids, which assign a separate roundness parameter to each axis, superquadrics share the same roundness for the x and y axes while allowing a distinct parameter for the z axis. Their shape is represented in implicit form by the equation:

$$f(\mathbf{x}) = \left(\left(\frac{x}{s_x}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{y}{s_y}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{z}{s_z}\right)^{\frac{2}{\epsilon_1}} = 1\,,$$

and the euclidean radial distance to their surface can be computed in closed form, as shown in Eq. 2. In addition, superquadrics are also equipped with an explicit function, which can be used to sample points from their surface. Specifically, given the coordinates (η, ω) such that $\eta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ and $\omega \in [-\pi, \pi]$, the surface of a superquadric can be represented as:

$$s(\eta, \omega) = \begin{bmatrix} s_x \cos(\eta)^{\epsilon_1} \cos(\omega)^{\epsilon_2} \\ s_y \cos(\eta)^{\epsilon_1} \sin(\omega)^{\epsilon_2} \\ s_z \sin(\eta)^{\epsilon_1} \end{bmatrix}.$$

We refer to [16] for additional details on superquadrics.

2. Training Details

In general, we train our model for 500 epochs with $\lambda_{par} = 0.1$ and then for other 500 epochs with $\lambda_{par} = 0.6$. For the results on Replica [45] and ScanNet++ [58] we trained our model on normalized objects (centered and rescaled to fit in a sphere of radius 0.5). During training we apply rotation and translation augmentations. Specifically, we apply random rotations between $0°$ and $180°$ around the z axis and between $0°$ and $7.5°$ on the x and y axis. We also apply random translation with respect to the center in a radius of 0.05. We optimize our network with Adam [26] a one-cycle learning rate schedule with a maximum learning rate of $4e-4$. We train on 4 NVIDIA A100 with total batch size of 128.

3. Additional Results

Does LM improve our final predictions? In our approach we use LM optimization as a post processing step. In this experiment (Fig. 11) we want to assess how a different number of LM optimization rounds affects the final predictions in terms of L2 Chamfer Distance. In order to evaluate this aspect, we report L2 loss after different numbers of LM optimization steps, evaluating both in-category and out-of-category. From this experiment we can notice two main aspects. Firstly, we see that it leads to larger improvements in the out-of-category rather than in the in-category one. This is probably due to the less accurate initial predictions of our feedforward model in this setting and it shows that our optimization step can be used to decrease the gap between in-category and out-category. Secondly, we see that even if

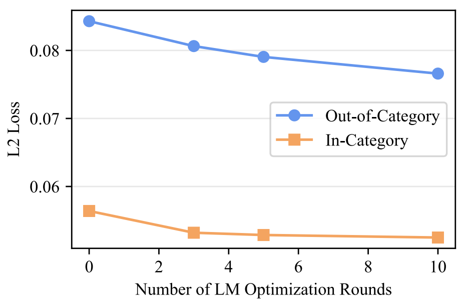

Figure 11. **LM optimization experiment.** We show how LM optimization improves results in terms of L2 Chamfer distance across a variable number of rounds. We report results both for in-category experiments and out-of-category ones.

Method	Training	Inference	L1 ↓	L2 ↓	# Prim.↓
SQ	-	-	3.668	0.279	10
SUPERDEC (Ours)	✓	✓	**1.698**	0.047	**5.79**
SUPERDEC (Ours)	✓	✗	**1.695**	**0.046**	5.82
SUPERDEC (Ours)	✗	✗	1.711	0.049	5.84

Table 4. **Ablation study on existence head.** Evaluation on objects from ShapeNet [4] dataset. The first two columns indicate whether we use the predicted α_j at training (left) and inference (right) time. Scores are scaled by 10^2.

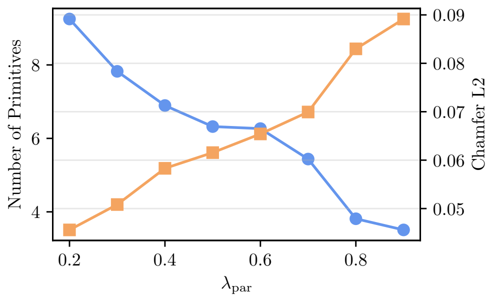

Figure 12. **Compactness vs. reconstruction accuracy tradeoff.** We run experiment for different values of the parsimony weight λ_{par} (x-axis) and we visualize the resulting number of primitives (y-axis, *left*) and the L2 Chamfer distance (y-axis, *right*).

		ScanNet++ [58]			Replica [45]		
Method	FPS	L1 ↓	L2 ↓	# Prim.↓	L1 ↓	L2 ↓	# Prim.↓
CSA	✓	2.91	0.41	11.64	3.68	0.70	9.63
SUPERDEC (Ours)	✓	**1.70**	**0.11**	**5.18**	**1.79**	**0.19**	6.58
SUPERDEC (Ours)	✗	1.74	0.13	5.19	1.81	0.20	**6.52**

Table 5. **Ablation study on sampling technique.** Evaluation on objects from 3D scene datasets [45, 58]. The second column report whether Farthest Point Sampling (FPS) was used to sample the input points. Alternatively, points were randomly sampled. Scores are scaled by 10^2.

LM optimization improves our final predictions, it does not lead to substantial improvements. This suggests that the solutions predicted by our method are located in local minima and that a diverse type of optimization should be resorted to improve the predictions further.

Is the existence head needed? As explained in the main paper, SUPERDEC predicts a set of P superquadrics and a segmentation matrix. As in many cases less than P superquadrics are in fact needed to represent the input point cloud, we also predict an existence parameter (α_j) for each of them, which allows us to model a variable number of primitives. However, the existence of a superquadric can also be directly deducted from the segmentation matrix, by computing $\hat{\alpha_j}$. We made some experiments to assess the impact of using either α_j or $\hat{\alpha_j}$, both at training and at inference time and reported the results in Tab. 4. In order to evaluate the impact of the prediction head, we conducted two additional experiments. In the first experiment, we evaluate the model trained as explained in Sec. 3.1. In the second experiment, we retrained our model without explicitly supervising α_j and using $\hat{\alpha_j}$ at inference time.

Compactness–Accuracy Trade-off. The hyperparameter λ_{par} controls the trade-off between reconstruction accuracy and representation compactness (see Eq. 4). We evaluate this trade-off quantitatively by first training the model with $\lambda_{par} = 0.1$ for 500 epochs, followed by fine-tuning for 100 epochs with varying λ_{par} values. Fig. 12 shows the impact of λ_{par} on Chamfer distance and the average number of predicted primitives. By adjusting λ_{par}, the model can smoothly balance compactness and accuracy, allowing for easy fine-tuning to meet target reconstruction quality. In our experiments, we use $\lambda_{par} = 0.6$, which approximately corresponds to the intersection point of the two curves.

Is FPS needed? To understand whether point clouds need to be downsampled with Farthest Point Sampling (FPS) or points can just be randomly sampled, we compare the performances of SUPERDEC on 3D scene datasets using the two approaches. We observe that the performances do not degenerate and that random sampling is enough for practical applications.

4. Qualitative results on scenes

4.1. Replica

In Fig. 13 we visualize the superquadric decompositions for some scenes from Replica [45]. The instance masks are predicted using Mask3D [41]. The points not belonging to any instance are visualized in gray.

4.2. ScanNet++

In Fig. 14 we visualize the superquadric decompositions for some scenes from ScanNet++ [58]. The instance masks the ground truth instance masks. The points not belonging to any instance are visualized in gray.

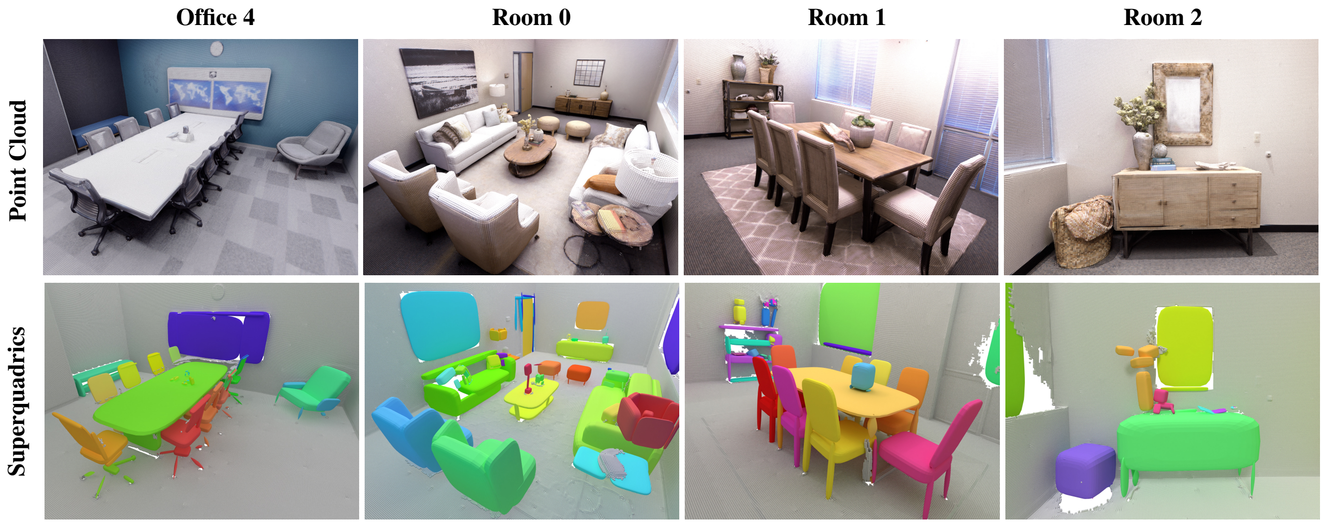

Figure 13. **Qualitative Results on Replica [45].** We show results on different scenes from Replica. In the first row we show a rendering of the input mesh, while on the second row we show the output superquadrics. We visualize different object instances with different colors.

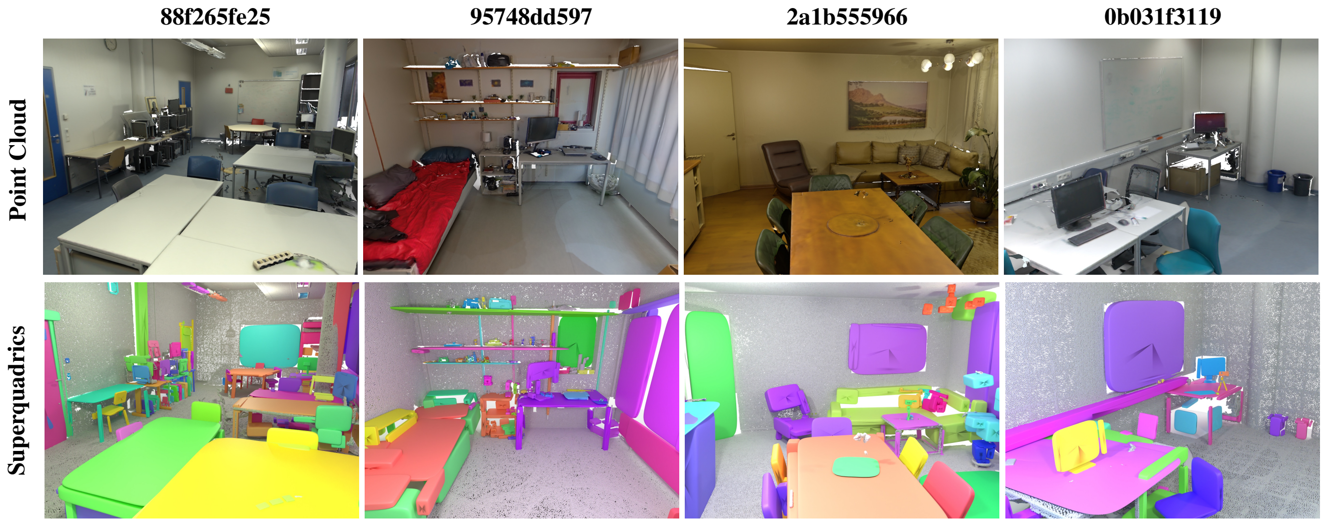

Figure 14. **Qualitative Results on ScanNet++ [58].** We show results on different scenes from ScanNet++. In the first row we show a rendering of the input mesh, while on the second row we show the output superquadrics. We visualize different object instances with different colors.

5. Robot Experiment

In this section we introduce the key methods and parameters used in our robot experiments. We also present more detailed qualitative and quantitative evaluation results.

5.1. Setup

For *path planning* in both ScanNet++ [58] and real-world scenarios, we use the Python binding of the Open Motion Planning Library (OMPL). The state space is defined as a *3D RealVectorStateSpace*, with boundaries extracted from the 3D bounding box of the input point cloud. We employ a sampling-based planner (RRT*), setting a maximum planning time of 2 seconds per start-goal pair.

In ScanNet++ scenes, the occupancy grid and voxel grid are both set to a 10 cm resolution, with voxels generated from the original point cloud. The collision radius is 25 cm. For dense occupancy grid planning, we enforce an additional constraint in the validity checking to ensure that paths remain within 25 cm of free space, preventing them from extending outside the scene or penetrating walls. And the planned occupancy grid path serves as a reference for computing relative path optimality in our evaluation. Start and goal points are sampled within a 0.4m-0.6m height range in free space, as most furniture and objects are within this range. This allows for a fair evaluation of how different representations capture collisions for valid path planning. During evaluation, we further validate paths by interpolating them into 5 cm waypoint intervals. Each waypoint is checked against the occupancy grid to ensure that its nearest occupied grid is beyond 25 cm and its nearest free grid is within 25 cm. A path is considered unsuccessful if more than 10% of waypoints fail this check. This soft constraint accounts for the sampling-based nature of RRT*, which does not enforce voxel-level validity but instead checks waypoints along the tree structure, leading to occasional minor violations. In the real-world path planning, we set the collision radius to 60 cm to approximate the size of the Boston Dynamics Spot robot. Spot follows the planned path using its Python API for execution.

For *grasping* in real-world experiments, we use the superquadric-library to compute single-hand grasping poses based on superquadric parameters. The process begins by identifying the object of interest and its corresponding superquadric decomposition. One of the superquadrics is selected and fed into the grasping estimator. To execute the grasp, the robot first navigates to the object's location during the planning stage. Then, using its built-in inverse kinematics planner and controller, the robot moves its end-effector to the estimated grasping pose for object manipulation.

5.2. Planning Results

In Tab. 6 we report the complete planning results on 15 Scannet++ [58] scenes.

Method	0a76e06478				0c6c7145ba				0f0191b10b				1a8e0d78c0				1a130d092a			
	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.
Occupancy	0.05	100	1.00	960KB	0.06	100	1.00	667KB	0.06	100	1.00	1031KB	0.05	100	1.00	926KB	0.05	100	1.00	803KB
PointCloud	0.07	86	0.98	18MB	0.09	91	0.99	12MB	0.03	77	0.99	19MB	0.05	91	0.99	18MB	0.05	89	0.98	18MB
Voxels	0.03	100	0.97	91KB	0.03	100	1.00	65KB	0.03	100	0.99	99KB	0.03	100	1.01	91KB	0.03	100	1.09	99KB
Cuboids [37]	0.11	32	0.98	22KB	0.10	18	1.02	19KB	0.14	85	1.03	34KB	0.10	50	1.06	21KB	0.12	79	1.00	27KB
SUPERDEC	0.17	100	0.99	52KB	0.16	100	0.97	48KB	0.17	92	0.94	51KB	0.14	91	0.99	39KB	0.13	100	0.98	35KB

Method	0a76e06478				0b031f3119				0dce89ab21				0e350246d4				0eba3981c9			
	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.
Occupancy	0.05	100	1.00	916KB	0.06	100	1.00	1760KB	0.05	100	1.00	1070KB	0.06	100	1.00	366KB	0.06	100	1.00	473KB
PointCloud	0.05	86	1.02	18MB	0.06	96	1.04	25MB	0.05	84	1.13	19MB	0.06	88	1.22	10MB	0.14	80	0.98	45MB
Voxels	0.03	100	1.01	99KB	0.03	100	1.00	160KB	0.03	100	1.19	104KB	0.03	100	1.00	51KB	0.03	100	1.12	199KB
Cuboid[37]	0.14	71	1.12	32KB	0.11	78	1.03	24KB	0.11	35	1.00	23KB	0.09	62	1.00	15KB	0.17	87	1.17	41KB
SuperDec	0.16	86	1.17	46KB	0.16	93	0.98	46KB	0.13	100	1.07	33KB	0.15	88	1.22	40KB	0.19	57	1.10	58KB

Method	7cd2ac43b4				1841a0b525				25927bb04c				e0abd740ba				0f25f24a4f			
	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.	Time(ms)	Suc.(%)	Opt.	Mem.
Occupancy	0.06	100	1.00	1241KB	0.05	100	1.00	1053KB	0.06	100	1.00	407KB	0.06	100	1.00	554KB	0.05	100	1	7MB
PointCloud	0.05	100	1.09	25MB	0.04	89	0.98	16MB	0.06	100	1.01	11MB	0.06	97	0.93	16MB	0.07	61	0.97	99MB
Voxels	0.03	100	1.00	137KB	0.03	100	0.98	82KB	0.03	83	1.04	51KB	0.03	100	1.04	83KB	0.03	96	0.96	617KB
Cuboid[37]	0.21	80	1.04	57KB	x	x	x	15KB	0.09	87	0.96	17KB	0.07	52	1.04	11KB	x	x	x	x
SuperDec	0.15	100	1.05	45KB	0.10	94	0.87	18KB	0.17	83	1.30	53KB	0.12	100	0.87	27KB	0.21	57	0.82	71KB

Table 6. **Path Planning Results.** We show results of path planning for different ScanNet++ [58] scenes, whose ids are reported on the top. PointCloud method uses dense point clouds from ScanNet++, all other methods process the same input point cloud. Time refers to average execution time of the validity-check function during the sampling stage of planning. Success rate (Suc.) is calculated after excluding trials where no representation could generate valid path due to randomness of start and goal sampling. The Cuboid method encounters an out-of-memory failure when fitting scene *0f25f24a4f* due to its large scale, and fails to find any valid path in scene *1841a0b525*.