SUPERDEC: 3D Scene Decomposition with Superquadric Primitives

Supplementary Material

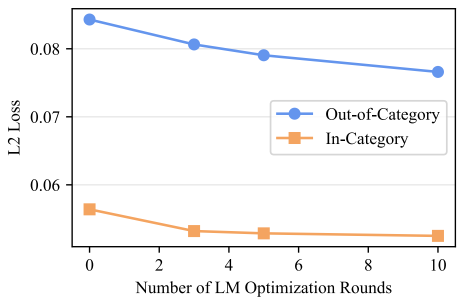

Figure 11. **LM optimization experiment.** We show how LM optimization improves results in terms of L2 Chamfer distance across a variable number of rounds. We report results both for in-category experiments and out-of-category ones.

6. 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 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.

Why 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.

7. 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.

7.1. Setup

For *path planning* in both ScanNet++ [56] 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.

7.2. Planning Results

In Tab. 4 we report the complete planning results on 15 Scannet++ [56] 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 4. **Path Planning Results.** We show results of path planning for different ScanNet++ [56] 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*.

From: **OpenReview** noreply@openreview.net
Subject: [ICCV 2025] Decision for Submission #4571
Date: 25 June 2025 at 10:46
To: efedele@ethz.ch

O

Dear Authors of Submission #4571,

Thank you for your submission to ICCV 2025. The review process has now concluded. Below you will find the meta-review and final reviews for your submission, which will be available on OpenReview shortly.

Congratulations, your submission #4571, titled “SuperDec: 3D Scene Decomposition with Superquadrics Primitives” has been accepted to ICCV 2025.
You will receive additional information for submitting the camera-ready version shortly. Please note that acceptance is contingent on passing a plagiarism check. Papers that do not comply with plagiarism and dual-submission rules will be rejected.

This year, we received 11,239 valid submissions that underwent the review process. The program committee recommended 2,698 papers for acceptance, resulting in an acceptance rate of 24%. All papers were initially evaluated by at least three independent reviewers. Following the author rebuttal and reviewer discussions, moderated by an Area Chair (AC), a triplet of three ACs reviewed each paper holistically, considering the reviews, author rebuttal, and reviewer discussions. For challenging cases, additional ACs and/or Program Chairs (PCs) were consulted.

We would like to thank the reviewers and Area Chairs for their contributions to the review process.

Best Regards,
ICCV Program Chairs