GECCO: Geometrically-Conditioned Point Diffusion Models

Michał J. Tyszkiewicz[1] Pascal Fua[1] Eduard Trulls[2]
[1]École Polytechnique Fédérale de Lausanne (EPFL) [2]Google Research, Zurich
michal.tyszkiewicz@epfl.ch pascal.fua@epfl.ch trulls@google.com

1. Experimental details

For simplicity we measure training in terms of gradient updates rather than dataset epochs.

Unconditional ShapeNet [7]. We use the download links provided by the PointFlow paper [7] directly. Their ShapeNet models contain 100k points: we subsample them to 2048 points while training, as explained in the paper. We train our models for 500k steps, with batch size of 64, and evaluate the model every 10k steps. For this evaluation we run a simplified variant of the benchmark described in section 4.1, which uses only the chamfer distance to pick the best checkpoint over the validation set – a variant of early stopping.

ShapeNet-vol [5]. We use the download links provided by the authors of OccNet [5], which contain prerendered images in addition to point clouds. Their point clouds are already subsampled to 2048 points. We train our models for 500k steps, with batch size of 48 and evaluate the model every 10k steps with a simplified version of the benchmark described in 4.2 not using ICP and with a random subset of the validation set. Since we observed no overfitting, we do not do early stopping.

Taskonomy [8]. The dataset comes in 4 sizes: *tiny*, *medium*, *full* and *full-plus*. We noticed that *full-plus* contains certain scenes with incomplete/faulty scans. We train our model for 1M steps on the *full* variant and evaluate it on a randomly sampled 1024-element subset of the 'validation' split every 10k steps - the subsampling is to save computation. Note that the monocular depth baseline of [6] was also trained (by the authors) on the Taskonomy dataset.

We use the same evaluation procedure as for ShapeNet-vol, this time with 8192 examples, and observe no overfitting, so we report numbers for the final checkpoint. This means that the results over 'validation' and over 'test' should be equivalent, up to uniformity in the dataset split, and report both in the paper.

However, we notice that performance is lower on the 'test' split, particularly in terms of the mean, and not the median. We trace this discrepancy to two out-of-distribution scenes in the 'test' subset: 'vacherie', a residential apartment with very high ceilings which GECCO misscales by a factor of 2×, and 'german', a gym filled with large windows and mirrors, resulting in ambiguous depth (see Fig. 3). Note that the monocular baseline predicts *relative* depth, which we then rescale using the ground truth, which makes it largely immune to such outliers. In fact, the large difference in performance between GECCO without ICP and with ICP is mostly explained by mis-estimating the scale of the scene, which the baseline doesn't suffer from, due to its use of ground truth scale. This is also likely the reason why the baseline performs worse if ICP estimates both scale and rotation – as explained in the paper, we disable scale estimation for it.

2. Unconditional evaluation metrics

Here we define the metrics of coverage (COV), minimum matching distance (MMD) and 1-nearest neighbour accuracy ($1 - \text{NNA}$). They are equivalent to those used in [7], but using a different notation. We assume n given reference point clouds from the test set $\mathbf{S_r} = \{p_1, ..., p_n\}$, n samples generated by the model $\mathbf{S_g} = \{q_1, ..., q_n\}$, and a distance function $D(\cdot, \cdot)$ which can be either CD or EMD (defined in main text).

Coverage considers the distances between $p \in \mathbf{S_r}$ and $q \in \mathbf{S_g}$ and measures the fraction of p which are the nearest neighbour of some q:

$$\mathbf{COV}(\mathbf{S_r}, \mathbf{S_g}) = \frac{1}{n} \left| \left\{ q \in \mathbf{S_r} : \left(\underset{p \in \mathbf{S_g}}{\exists} q = \underset{\hat{q} \in \mathbf{S_r}}{\arg\min} D(\hat{q}, p) \right) \right\} \right|. \quad (1)$$

Mean matching distance measures the average distance between a sample and its nearest reference point

$$\mathbf{MMD}(\mathbf{S_r}, \mathbf{S_g}) = \frac{1}{n} \sum_{p \in \mathbf{S_r}} \min_{q \in \mathbf{S_g}} D(p, q). \quad (2)$$

1-nearest neighbour accuracy measures the accuracy of a classifier distinguishing between elements of $\mathbf{S_r}$ and $\mathbf{S_g}$ by returning the class of the nearest example in $\mathbf{S_r} \cup \mathbf{S_g} - \{u\}$, where u is the element under consideration itself. We define this formally by introducing the asymmetric $H(\mathbf{S_a}, \mathbf{S_b})$ which counts the $a \in \mathbf{S_a}$ correctly classified w.r.t. the union of $\mathbf{S_a}$ and $\mathbf{S_b}$ and then define $\mathbf{1-NNA}$ as the average of H applied in both directions. We introduce $\mathbf{S_a}, \mathbf{S_b}, u$ and v to emphasize that in the final equation H is applied in both directions.

$$H(\mathbf{S_a}, \mathbf{S_b}) = \left| \left\{ u \in \mathbf{S_a} : \left[\underset{v \in (\mathbf{S_a} \cup \mathbf{S_b} - \{u\})}{\arg\min} D(u, v) \in \mathbf{S_a} \right] \right\} \right| \tag{3}$$

$$\mathbf{1-NNA}(\mathbf{S_r}, \mathbf{S_g}) = \frac{1}{2n} \left(H(\mathbf{S_r}, \mathbf{S_g}) + H(\mathbf{S_g}, \mathbf{S_r}) \right) \tag{4}$$

We note that EMD is usually approximated due to its computational cost, and that CUDA-based implementations may vary drastically; we use PyTorchEMD.

3. Additional visualizations

We invite the reader to view the video included in this supplementary material for conditional and unconditional samples from ShapeNet, as well as for image-conditional samples from Taskonomy. On the following pages of this document, in Fig. 1 we show additional unconditional samples from models trained on the ShapeNet, compared with baselines [4, 1]. In Fig. 2 we show additional samples from the image-conditional model trained on the Taskonomy dataset, along with the results of their upsampling. Note that we upsample the point clouds with the technique introduced in sec. 4.4 of the main paper, and how showcase it in the image-conditional case. Due to performance issues with the point cloud renderer we render the results upsampled only by 5x, but GECCO itself works equally well with 100k points, as shown in the main paper.

4. Efficient upsampling

During upsampling by inpainting, for optimal performance, it is preferable to keep the network's input distribution as close as possible to that at training time. This means that ideally, we would want $m = n + 1$, where n and m are the cardinality of the original and upsampled point clouds, respectively. Note that a cloud can be upsampled by a factor of f, to $(1 + f)n$ points, by repeating that procedure $f \times n$ times and concatenating the results. This however is as expensive as generating a cloud of n points $f \times n$ times from scratch.

Fortunately, the Set Transformer [3] architecture enables a workaround which brings the cost down to that of sampling a cloud of $f \times n$ points once. Since the points do not interact directly with each other, but rather via the inducers (see the main paper and [3] for definition), if we make the assumption that in the limit of large n the influence of any single point on the inducers is negligible, we can reverse-diffuse many new (inpainted) points *in parallel*, with shared inducer state. This leads to a simple algorithm starting from the conditioning set $\{p\}_{1..n}^{\sigma=0}$ and the pure noise inputs $\{p\}_{n..m}^{\sigma=\sigma_{\max}}$. At each step t of reverse diffusion:

1. The conditioning input is diffused to σ_t to obtain $\{p\}_{1..n}^{\sigma_t}$.
2. The score network is evaluated on $\{p\}_{1..n}^{\sigma_t}$. The score estimate is discarded and instead we cache the activations of the inducers across the network's layers.
3. The score network is ran again, this time on $\{p\}_{n..m}^{\sigma_t}$, but with the inducer activations provided by the cache from point 2. We obtain the score for the inpainted points $n..m$.
4. We update $\{p\}_{n..m}$ as usual, using the score from point 3.

The results provided in the main paper and here are obtained by the naive procedure, upsampling multiple times with a context of $\frac{1}{2}n = 1024$ but we also release code implementing the improved scheme described here.

5. Dataset licenses

ShapeNet. The original ShapeNet [2] dataset of 3D meshes is licensed for non-commercial use, with clauses allowing for redistribution of derived assets. Our use of derived datasets released with [7] and [5] fulfills these provisions.

Taskonomy. Our dataset of real world imagery is derived from the data released with Taskonomy [8], which allows non-commercial use in its license. Bound by the license, we are not able to release our derivatives until authorized by the authors of [8]. Depending on their permission, we intend to release either our dataset or the code to re-generate it from the original files Taskonomy files.

6. Emoji license

In the video material we use an emoji (🦎) from the Noto emoji bank, licensed under Apache 2.0 license.

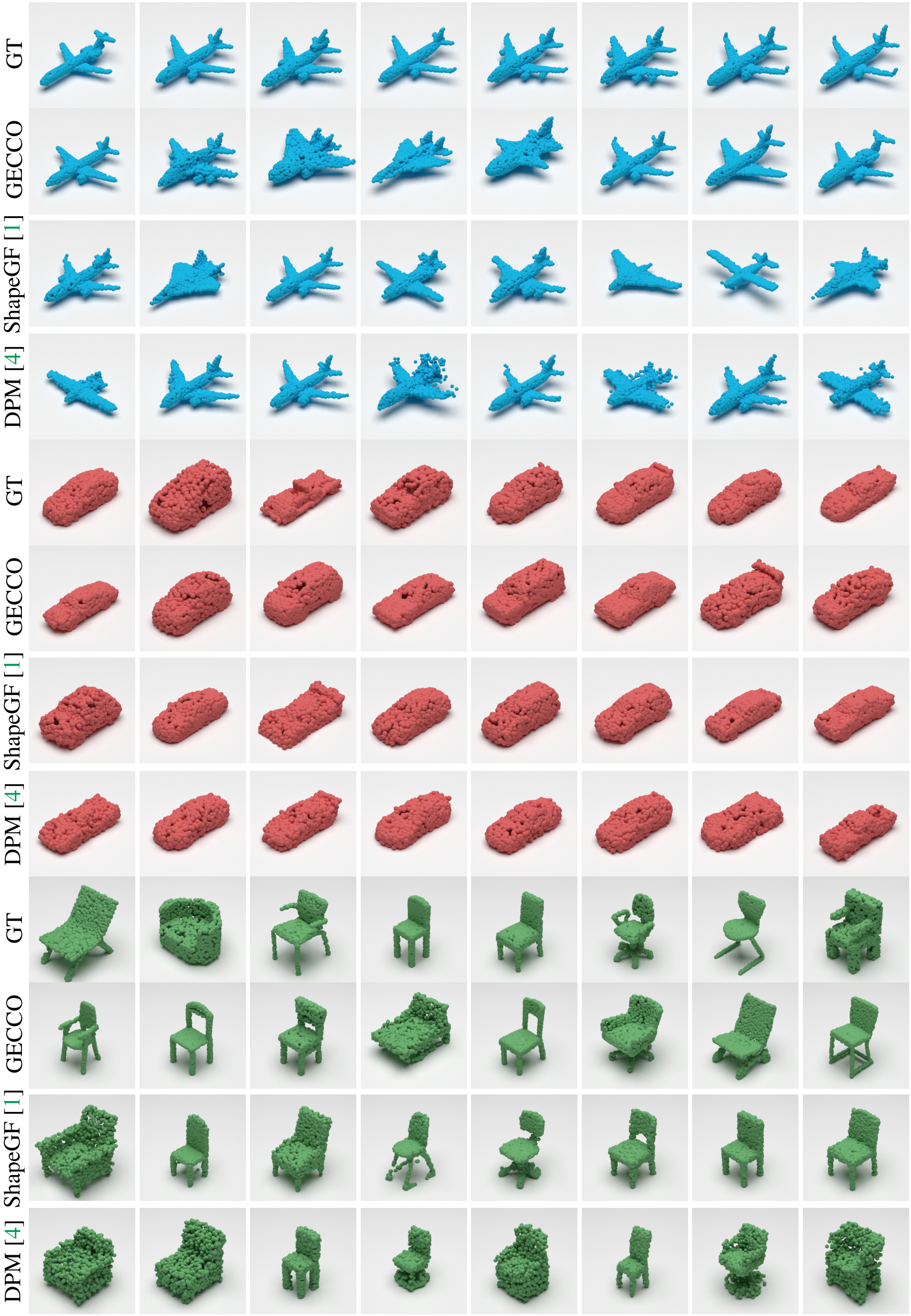


Figure 1: **Unconditional point cloud synthesis on ShapeNet.** All examples contain 2048 points.

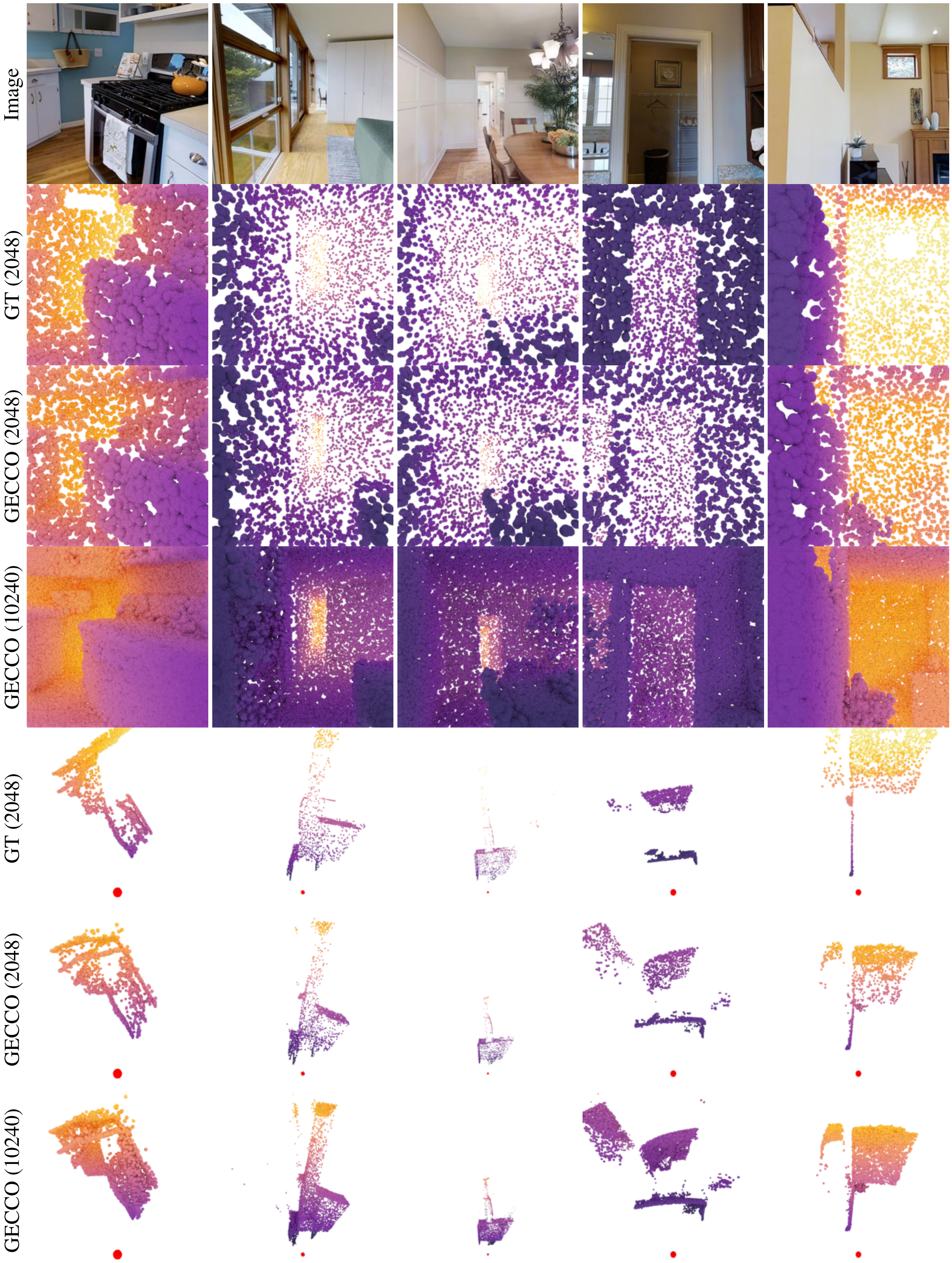
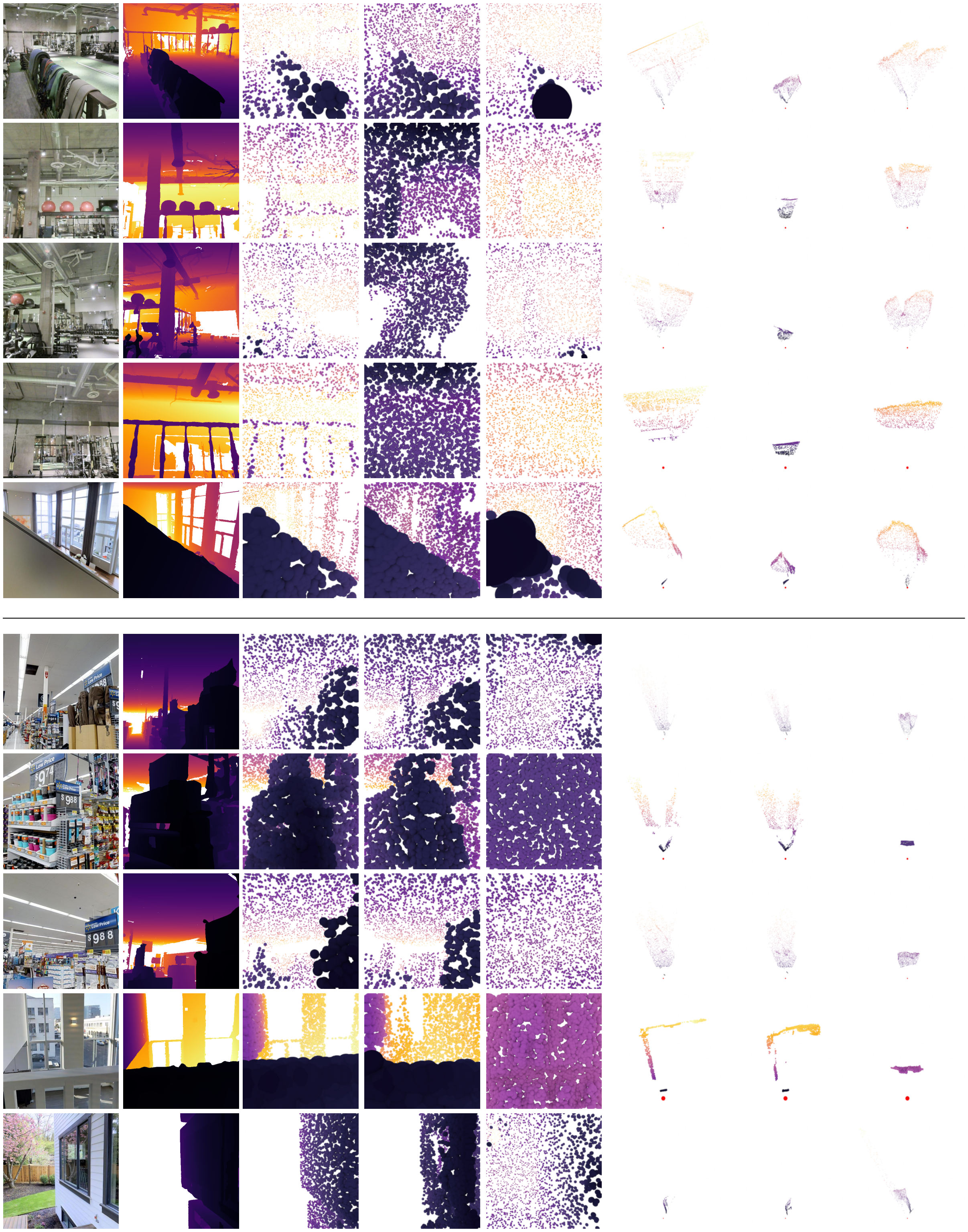


Figure 2: **Image-conditional point cloud synthesis and upsampling on the test set of Taskonomy.** We render the point clouds with smaller point radii, to highlight the difference made by upsampling. Note how in column 4 GECCO is tricked by a mirror, on the left side.



(a) Source image (b) GT depth (c) GT POV (d) GECCO POV (e) [6] POV (f) GT BEV (g) GECCO BEV (h) [6] BEV

Figure 3: **Outliers in Taskonomy.** We showcase the top 5 test images where the difference in chamfer distance is most in favor of the monocular baseline (top) and GECCO (bottom). GECCO suffers from mirrors (top 4 rows, 'german') and unusual room dimensions (last row, 'vacherie'), while the baseline cannot resolve areas of undefined depth, such as in the two bottom rows.

References

[1] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *European Conference on Computer Vision*, pages 364–381. Springer, 2020. 2, 3

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[3] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019. 2

[4] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021. 2, 3

[5] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[6] Alexander Sax, Jeffrey O Zhang, Bradley Emi, Amir Zamir, Silvio Savarese, Leonidas Guibas, and Jitendra Malik. Learning to navigate using mid-level visual priors. In *Conference on Robot Learning*, pages 791–812. PMLR, 2020. 1, 5

[7] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. 1, 2

[8] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 1, 2