Supplemental Material for: SHRED: 3D Shape Region Decomposition with Learned Local Operations

R. KENNY JONES, Brown University, USA
AALIA HABIB, Brown University, USA
DANIEL RITCHIE, Brown University, USA

ACM Reference Format:
R. Kenny Jones, Aalia Habib, and Daniel Ritchie. 2022. Supplemental Material for: SHRED: 3D Shape Region Decomposition with Learned Local Operations. *ACM Trans. Graph.* 41, 4, Article 186 (July 2022), 5 pages. https://doi.org/10.1145/3550454.3555440

A SHRED IMPLEMENTATION DETAILS

A.1 Synthetic Data Creation

Fix Network. To generate an input-output training example for the fix network, we employ the following procedure. From our training dataset, we first sample a random region from a random shape, using the ground-truth fine-grained part annotations to define a region. We increase the size of the region by Y% (Y between 5 and 25) with a 75% chance: this involves sampling a random point about the regions center, finding the Y% closest points currently outside the region, and flipping their sign. By a similar procedure we randomly remove between 10 and 50% of the inside region points with a 75% chance. Next we up or down sample the inside region points, and nearby outside points (within a 0.1 extended radius of the region's center + radius) to form two 2048 point clouds. Each point in this point cloud has its label flipped with some Z% (Z randomly set to be between 0.0 and 0.3). To offset data imbalance that causes more inside labels to flip to outside, we finally randomly flip any extra inside points (those down-sampled in the previous step) to become outside points with a 10% probability. This allows the network to see more cases where outside points become inside points during training. To decide whether to keep or reject the entire sample, we calculate the percentage of input inside points that have inside as their target label, and the percentage of inside label points that are associated with an inside input point: if either of these percentages is below 40%, we skip the example.

Merge Network. To generate training data for the merge network, we employ the following procedure. First we sample a random shape from the training dataset. Then we use FPS to split the shape into M regions (M randomly chosen from 16, 32, 64, 128). For each region, we first identify all ground-truth part instances that appear in the region. We split each of these regions into K sub-regions, where K is sampled between 1 and 10 according the probability distribution proportional to 0.5^K. To execute this split, we sample K points randomly distributed about the center of the region, and then assign each point in the region to the randomly sampled point it is closest to. Each sub-part is then assigned to some cluster of regions randomly: with 1/3 chance it is merged into a default group with other sub-parts in the same FPS defined region, with 1/3 chance it is merged into one of K groups with other sub-parts in the same FPS defined region, and with 1/3 chance it is merged into one of K groups with other sub-parts in the same FPS defined region from the same GT part instance.

At this point, we are ready to start gathering merge examples. We first find all neighboring regions from our above synthetic split procedure. We then randomly sample a pair of neighboring regions, and check which ground-truth parts they most heavily overlap with. If they overlap most with the same ground-truth part, then we record the pair as an example where a split *should* happen, otherwise we record the pair as an example where a split *should not* happen. To produce merge examples at different granularities, we also modify the regions during this procedure; every time a merge should happen, we actually perform the merge 75% of the time, and every time a merge should not happen, we actually perform the merge 25% of the time. We continue collecting merge examples until we run out of neighbor pairs that have not been previously considered.

B SHRED METHOD ABLATIONS

B.1 Modified Hungarian Matching Algorithm

The split network is trained with a cross entropy loss between predicted instances and target instances. Unlike most settings where cross entropy is used, for the instance segmentation task there is not a canonical mapping from prediction slots to target slots. For this reason, a typical approach has been to dynamically use the Hungarian matching algorithm to find a one-to-one mapping between prediction slots and target slots that would minimize the cross entropy loss. For an example, see [Mo et al. 2019].

While this matching scheme is optimal when an exact instance-segmentation is desired, we would prefer that the split network removes *all* under-segmentation, even if this comes at the cost of producing more regions. As our merge module will later figure out how to combine different over-segments, we want the split network to solve an over-instance segmentation task. To this end, we modify the matching procedure between prediction slots and target slots to greedily search for opportunities to reward network over-segmentation predictions.

Authors' addresses: R. Kenny Jones, Brown University, USA; Aalia Habib, Brown University, USA; Daniel Ritchie, Brown University, USA.

The procedure takes as input a logit prediction over N points with K slots P, a $(N\ x\ K)$ matrix, and a ground-truth instance segmentation T, a $(N\ x\ K)$ matrix where each column is a one-hot vector. To begin, we employ the typical Hungarian matching approach to find a one-to-one assignment M between prediction and target slots. We identify all unassigned prediction and target slots (e.g. those that correspond to empty instances). Then for each paired unused prediction slot, U_P, and unused target slot, U_T, we find all indices in P where U_P was the argmax prediction. We then index into T with these indices, and find the mode region in T that best aligns with U_P, call this region A. Next, we find the slot of P that was assigned to A under M, call this slot P_A. We find the index rows of T where A is set to 1.0, call these T_A. Then we use the logit predictions of P_A and U_P to split T_A into two parts, T_{PA} and T_{UP}, with an argmax operation.

At this point we have identified a possible over-segmentation: we could modify T_A to become split into two parts, T_{PA} and T_{UP}, where prediction slot P_A would be re-assigned to the region T_{PA} in slot A and unused prediction slot U_P would be assigned to region T_{UP} in slot U_T. If T_{PA} and T_{UP} both contain more than 10 indices each, and A was the argmax prediction for more than 50% of the indices under U_P, then we accept this over-segmentation with modifications to T and M. This procedure is repeated for ever pair of unused slot.

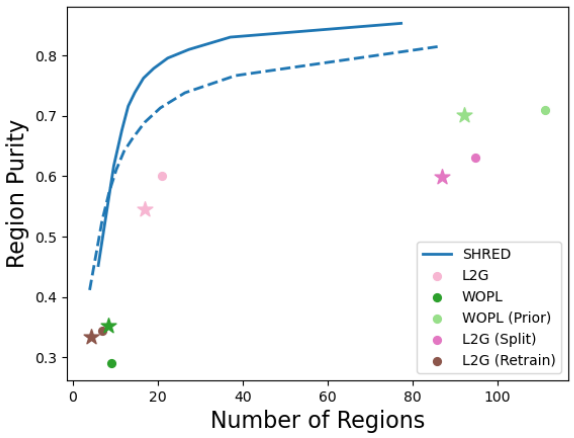

Fig. 6. Trade-off between region granularity and region quality for additional baseline conditions. WOPL and L2G contain intermediary stages that correspond with SHRED's split stage, so we plot the results of these stages as separate conditions (WOPL Prior and L2G Split). Our attempt to retrain L2G on our training dataset failed to converge to good performance (brown dot).

B.2 Naive Synthetic Data Creation

In Section 4.6 of the main paper, we present results of SHRED when learned local operators are replaced with versions trained under naive synthetic data creation (SDC) procedures. In this section, we describe the naive SDC procedures used for the split, align and merge modules for each ablation condition.

Naive split SDC. To generate naive training data for the split network we employ the following procedure. Given a training shape, we first generate a region decomposition by iteratively: (1) sampling a random point from the shape, (2) sampling a random radius r, from a beta distribution with alpha as 1.5 and beta as 4.0, (3) assigning all points within r radius of the randomly sampled point to a new region (with a limit of up to half of the remaining unassigned points). If we want to produce K regions, we run this iterative procedure K-1 times, then throw all unassigned points into the Kth region (we set K=64). Once a region decomposition has been produced, we follow the logic of the default SDC, where each region in the naive decomposition contributes one input point cloud for training, and ground truth target instances are supplied by the fine-grained annotated labels.

Naive fix SDC. To generate naive training data for the fix network we employ the following procedure. From our training dataset, we first sample a random surface point from a random shape. Then we sample a random radius r from the same distribution as in the Naive split SDC case. All points with r radius of the randomly sampled point are then set to be within the target region (up to 25% of the points in the shape). Following the default procedure, we then up or down sample the inside region points, and nearby outside points (within a 0.1 extended radius of the region's center + radius) to form two 2048 point clouds. We then randomly flip the label of each point, with a probability of 15%. The target inside-outside values of this example are then computed by finding the region in the ground-truth labeling that has the maximum overlap with the inside points of the example.

Naive merge SDC. To generate naive training data for the merge network we employ the following procedure. First a naive region decomposition is produced in the same manner as described in the *Naive split SDC* paragraph (where K is randomly chosen from the set of 16, 32, 64, 128). Then, we generate merge examples following the default SDC logic, sampling random merges, recording whether they should happen with respect to the ground-truth region annotations, and randomly actualizing the merge (full details in the last paragraph of Section A.1).

C BASELINE IMPLEMENTATION DETAILS

C.1 ACD

We use the VHACD implementation from [Mamou 2016]. This implementation is widely used by a multitude of applications, and has been used by related work to formulate self-supervised training objectives [Gadelha et al. 2020]. We found this ACD implementation works best when input meshes are manifold; as such we do not directly apply it to PartNet meshes, as many are double-sided or contain inconsistent normal orientations. Instead, we first send each PartNet mesh through a manifold procedure [Huang et al. 2018], and run the VHACD algorithm on the manifold scipt output. ACD generated convexes are then used to partition a high-resolution point cloud (100k points) into segments by sampling the surface of each convex and finding the nearest neighbor from each point in the high-resolution point cloud to the point cloud of convexes.

C.2 PN Seg

We retrain Partnet's instance segmentation model using our training data and the author's publicly released code at https://github.com/daerduoCarey/partnet_seg_exps .

Following [Luo et al. 2020], we remove the semantic label prediction head and semantic label loss, as we train over multiple categories at once. We follow all other default hyper-parameters as found in the code. We write a procedure to export our training data into the h5 file format that their method expects, and then start training runs through their released script.

C.3 L2G

L2G trains a version of their method on the same in-domain categories that SHRED uses, but uses slightly more data for each category [Luo et al. 2020]. For all experiments in the main paper, we use the author's released models from https://github.com/tiangeluo/Learning-to-Group . While this causes the training distribution between SHRED and L2G to differ slightly, if anything we believe this hurts SHRED, as it has access to less data. The L2G method has an initial step before merging where a simple model produces a shape over-segmentation; we show how this stage fairs on the trade-off graph between decomposition granularity and purity in Figure 6 (dark pink points).

Using the released code, we attempted to retrain a version of L2G on the subset of chair, lamp, and storage data used to train SHRED. We converted our data into the same h5 file as we used for PN Seg, and then initiated L2G training with their specified scripts, keeping all method hyper-parameters unchanged. Interestingly, we found that this version of L2G converged to much worse performance, we plot this retrained version of L2G on Figure 6 (brown points).

C.4 WOPL

The WOPL method has no publicly available code and we were unable to gain access to an implementation by contacting the authors. To this end, we attempted to implement the algorithm as described in the paper [Wang et al. 2021]. We include our implementation of the WOPL method in the released code, where in all cases we tried to follow the paper's description as closely as possible. The WOPL method has an analogous stage to SHRED's split stage, termed the prior stage, that creates a region decomposition before a global merge step. In Figure 6 we show the trade-off the WOPL prior makes between decomposition granularity and quality (light green points).

D SEMANTIC SEGMENTATION EXPERIMENT DETAILS

Our experiments in Section 4.5 demonstrate how SHRED can benefit fine-grained semantic segmentation methods when training data with semantic labels is limited. We use SHRED to generate region decompositions that are passed into NGSP: an approach that learns to assign semantic labels from a fine-grained grammar to regions of a 3D shape.

We evaluate SHRED (and other region decomposition approaches, see PN Seg, L2G, and ACD rows of Table 2 in the main paper) under two limited labeled data paradigms: when 10 or 40 shapes with semantic labels are provided to train a semantic segmentation method. We first use the region decomposition methods (trained on all in-domain categories as described in sections 4.1 and 4.2) to find a region decomposition for the 10 or 40 training shapes that have semantic label annotations. With a training dataset of these shapes that contain both semantic annotations and region decompositions, we then have the data needed to train NGSP for a given category.

NGSP trains separate networks for each category, as semantic labeling is specific to a category. We follow NGSP's procedure to train a guide network and likelihood networks for each category [Jones et al. 2022]. For our experiments in Table 2 of the main paper, we use the 5 categories studied by both SHRED and NGSP: chairs, lamps, storage, tables, and knives. Guide network training is relatively inexpensive. A guide network can be trained in less than 10 minutes when the number of labeled shapes is low (the settings we are interested in), so we retrain a separate guide network for each category and for each region decomposition method (e.g PN Seg, L2G, ACD, SHRED). Likelihood network training is more involved, as it requires training multiple networks for each node in a fine-grained semantic grammar (as reference, the chair grammar has over 30 nodes). Therefore, we train a single collection of likelihood networks that are shared by all region decomposition methods; the likelihood networks use ground-truth region annotations during training (as we assume each shape with semantic annotations also has part instance annotations).

Once the guide and likelihood networks have been trained, we populate the values of Table 2 with the following procedure. A input shape for category C is first decomposed into a set of regions with region decomposition method M. The guide network specialized for the (C, M) pair is then used to propose a set of 10000 label assignments over the region decomposition. The likelihood network specialized for C then evaluates all proposed label assignments, and chooses the label assignment that maximizes equation 1 from the NGSP paper. Note that we also include the guide network likelihood into this equation, finding it regularizes the predictions of the likelihood networks that have never seen region decompositions produced by M. Per-region semantic label predictions can then be propagated to the points within each region using the chosen label assignment. Finally, the semantic mIoU can be calculated by finding the intersection over union between predicted and ground-truth per-point labels, and averaging this value over the labels of the grammar across the test-set shapes of each category.

The No Reg rows in Table 2 and 3 in the main paper correspond with predictions from the fine-grained semantic segmentation network proposed by [Mo et al. 2019]. This network operates globally, and learns to label individual points instead of shape regions.

E ADDITIONAL QUALITATIVE RESULTS

We provide additional qualitative comparisons between the region decompositions produced by various methods in Figure 7. Columns 6-8 show how SHRED's predictions change as the merge-threshold hyper-parameter is modified (we look at the values of 0.2, 0.5, and 0.8).

REFERENCES

Matheus Gadelha, Aruni RoyChowdhury, Gopal Sharma, Evangelos Kalogerakis, Liangliang Cao, Erik Learned-Miller, Rui Wang, and Subhransu Maji. 2020. Label-Efficient

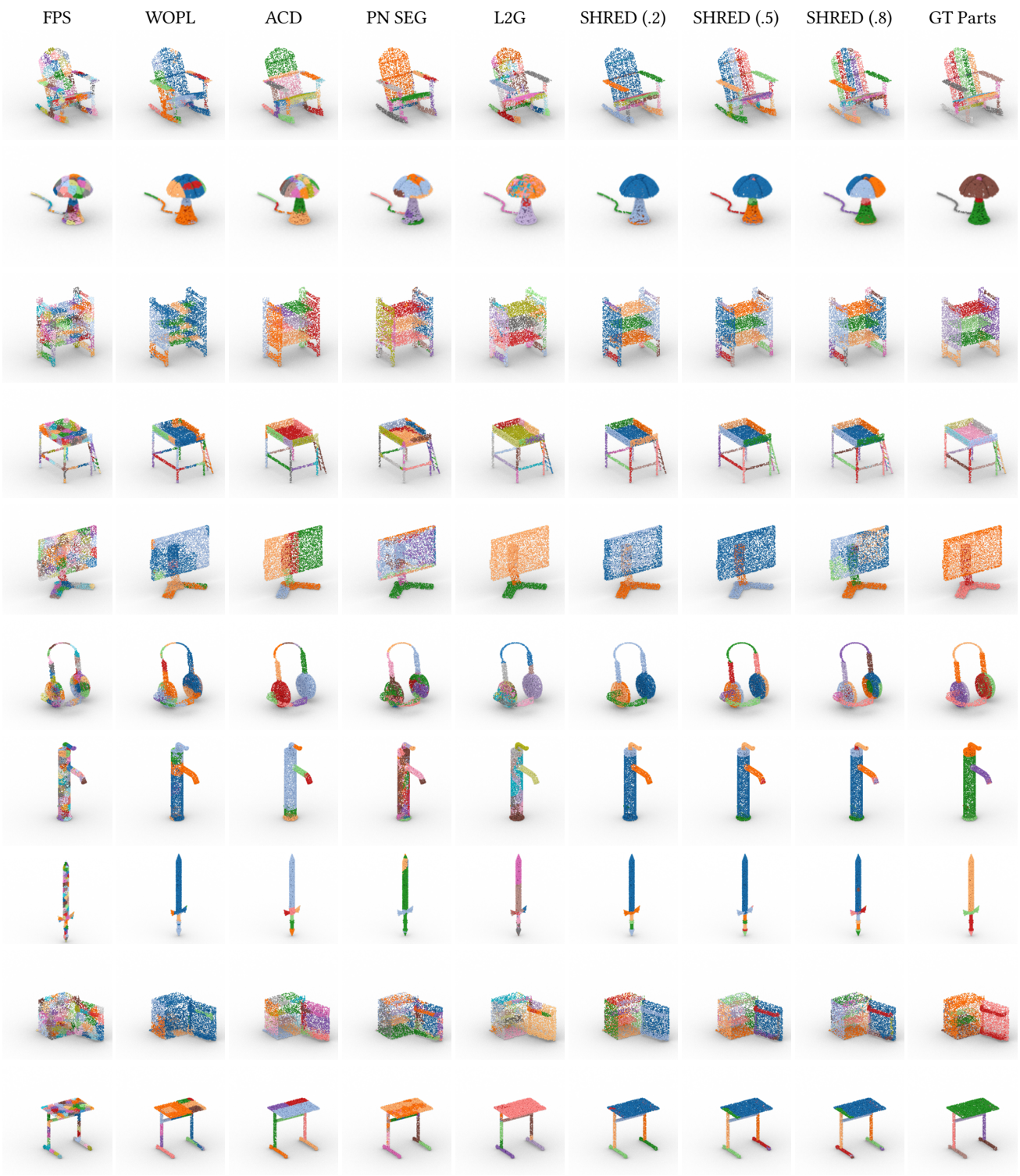

Fig. 7. Additional qualitative region decomposition comparisons between SHRED and baseline methods. In columns 6-8, we show how varying SHRED's merge-threshold changes the granularity of the output decomposition.

Learning on Point Clouds using Approximate Convex Decompositions. In *European Conference on Computer Vision (ECCV)*.

Jingwei Huang, Hao Su, and Leonidas Guibas. 2018. Robust Watertight Manifold Surface Generation Method for ShapeNet Models. *arXiv preprint arXiv:1802.01698* (2018).

R. Kenny Jones, Aalia Habib, Rana Hanocka, and Daniel Ritchie. 2022. The Neurally-Guided Shape Parser: Grammar-based Labeling of 3D Shape Regions with Approximate Inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Tiange Luo, Kaichun Mo, Zhiao Huang, Jiarui Xu, Siyu Hu, Liwei Wang, and Hao Su. 2020. Learning to Group: A Bottom-Up Framework for 3D Part Discovery in Unseen Categories. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkl8dlHYvB

Khaled Mamou. 2016. Volumetric Hierarchical Approximate Convex Decomposition. In *Game Engine Gems 3*, Eric Lengyel (Ed.). A K Peters, 141–158.

Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. 2019. PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiaogang Wang, Xun Sun, Xinyu Cao, Kai Xu, and Bin Zhou. 2021. Learning Fine-Grained Segmentation of 3D Shapes Without Part Labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10276–10285.