Supplemental Material: Self-Supervised Representation Learning for CAD

Benjamin T. Jones[1] Michael Hu[1] Milin Kodnongbua [1] Vladimir G. Kim[2] Adriana Schulz[1]
[1]University of Washington [2]Adobe Research
{benjones, mkhu, milink, adriana}@cs.washington.edu vokim@adobe.com

A. Representation Learning Architecture

Input Encodings. Each topological entity is initially encoded as a vector encoding its parametric geometry and that geometry's relationship to the topological entity. The three dimensions of entity; face, edge, and vertex, each have their own unique encoding of their own size, illustrated in Figure 1.

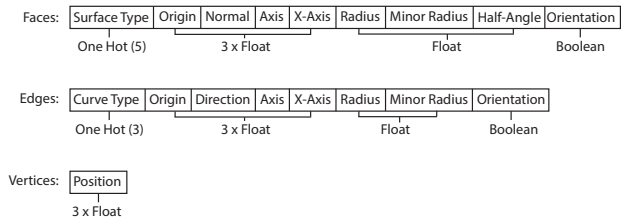

Figure 1. Input encodings for the 3 different topology types. Not all surface and curve types use all of the available parameters; unused parameters are encoded as zeros.

We want a common, fixed size for each level of topology input vetor, so we limit our input to B-Reps that have geometry with a fixed number of parameters: planes, cylinders, cones, spheres, and tori for surfaces, and lines, circles, and ellipses for edges. We validated this choice by filtering the parts in the Fusion 360 Segmentation and Automate datasets [3, 4] and found that 77% and 72%, respectively, contained only these primitives.

Surfaces types use the following parameters:

- **Plane**: Origin, Normal, X-Axis
- **Cylinder**: Origin, Axis, X-Axis, Radius
- **Cone**: Origin, Axis, X-Axis, Radius, Half-Angle
- **Sphere**: Origin, Axis, X-Axis, Radius
- **Torus**: Origin, Axis, X-Axis, Radius, Minor Radius

Orientation is common to all faces, and is a boolean value indicating if the surface normal is parallel or anti-parallel to the face normal.

Curve types use the following parameters:

- **Line**: Origin, Direction, X-Axis
- **Circle**: Origin, Axis, X-Axis, Radius
- **Ellipse**: Origin, Axis, X-Axis, Radius, Minor Radius

Orientation is common to all edges, and is a boolean value indicating if the curve parameterization is the same as, or reversed from, the edge's. This is important because curve orientation is necessary to determine the inside versus outside of bounded faces.

We re-parameterize plane, cylinder, and line geometries so that the origin coordinate is as close to the coordinate system origin as possible. This is done because CAD programs often choose origins far from the actual geometric position. Prior work has dealt with this by omitting these parameters [3].

Face, edge, and vertex input encodings are collected into 3 input matrices; $F^{(0)}$, $E^{(0)}$, and $V^{(0)}$ for use in an adjacency list representation of the B-Rep graph. Two adjacency lists are used to represent the multi-partite graph; $\overrightarrow{VE}$ mapping vertices to the edges they bound, and $\overrightarrow{EF}$, mapping edges to faces they bound. Edge graph adjacency has an associated boolean *orientation* feature, $O(e)$, $e \in \overrightarrow{VE} \cup \overrightarrow{EF}$. For $\overrightarrow{VE}$ this indicates if a vertex is a start or end point of the associated edge, and for $\overrightarrow{EF}$ it represents if the inside of the associated face is on the left or right of the edge, relative to the edge's parameterization direction.

Encoder. We structure our encoder as a hierarchical message passing network, inspired by the upwards pass of SB-GCN, but using graph attention message passing with edge features (adapted from [7] and implemented in PyG [2]), allowing us to incorporate the adjacency features, as well as omit the input MLPs used by SB-GCN. The full encoder architecture is shown on the left half of Figure 2.

Decoder. Our decoder is modeling a function that maps (u, v) coordinates to (x, y, z) positions plus a clipping mask SDF d conditioned on a face code $f \in F^{(1)}$. We parameterize this as a 4-layer, fully connected ReLu network similar

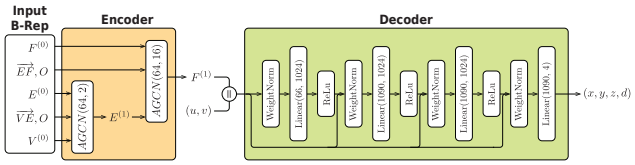

Figure 2. Our geometric self-supervision encoder and decoder architecture. $\|$ denotes concatenation, and $AGCN(S, H)$ is a graph message passing layer with embedding size S and H-headed attention. We use $H = 2$ attention heads for $\overrightarrow{VE}$ since edges have at most two vertices, and $H = 16$ for $\overrightarrow{EF}$ because we observed greater performance with many attention heads, and 16 was an empirically determined balance between model size and accuracy.

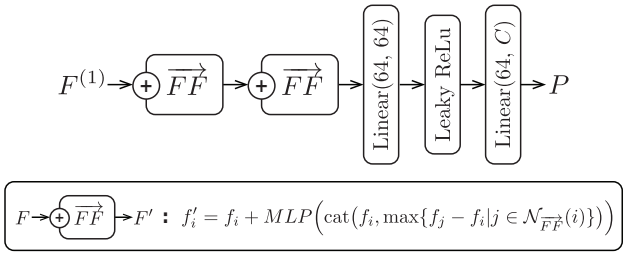

Figure 3. Few-Shot segmentation architecture. C denotes the number of classes, and P the output logits. In the inset, f_i and f_j denote rows of the feature matrix F, and $\mathcal{N}_{\overrightarrow{FF}}(i)$ denotes the neighbors of node i in adjacency list $\overrightarrow{FF}$.

to that of DeepSDF [6], as shown on the right side of Figure 2.

B. Few-Shot Learning Architectures

Segmentation Our segmentation architecture takes as input the learned face embeddings $F^{(1)}$ from our representation learning, and the B-Rep face-to-face adjacencies $\overrightarrow{FF}$, and consists of two Residual MR-GCN graph message passing layers [5], followed by a 2 layer LeakyReLu MLP, pictured in Figure 3.

Classification Our classification architecture takes as input the learned faces embeddings $F^{(1)}$ and consists of two linear layers separated by a max-pool and LeakyReLu, pictured in Figure 4.

C. Biased SDF Sampling

To facilitate training of the neural implicit part of our geometric self supervision, we want to preferentially sample each face's parameter space near clipping mask boundaries, and to approximate the distance to boundaries. Neither of these operations are supported natively by CAD kernels (OpenCascade in our case [1]). They do, however, support efficiently querying if a point in a surface parameterization is inside or outside the clipping function. Therefore, we approximate the SDF by sampling a large number of uv-points (5000) and matching each inside and outside point to its nearest neighbor in the opposite set using a KD-Tree to accelerate these queries. We keep $N = 500$ of these points on each face for training. To bias our sample towards the 0-level set, as is common when training implicit SDFs [6], we sort by $|d|$, task 40% of our sample to be the points nearest to the boundary, and randomly sample the rest.

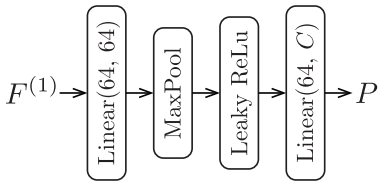

Figure 4. Few-Shot classification architecture. C denotes the number of classes, and P the output logits.

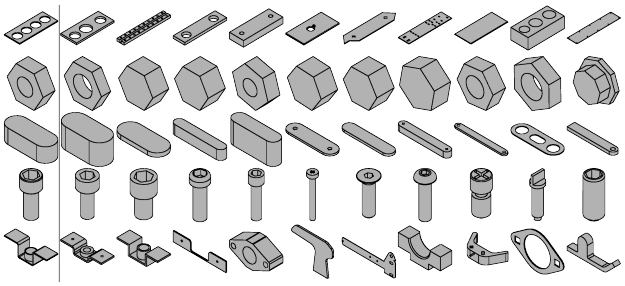

Figure 5. Unsupervised part retrieval on Fusion 360 Segmentation Test set. Query on left, 10 nearest neighbors, sorted from closest to furthest on right.

D. Unsupervised Part Retrieval

Although our embeddings are computed per face rather than per-part, we have found that the face embeddings are still useful for part-level retrieval. We max-pooled face embeddings across each part to form a part embedding and used these to find nearest neighbors with the Fusion 360 Segmentation Test set. **??** shows the 10 nearest neighbors for a diverse collection of part queries, showing qualitatively that face-level embeddings can be used to search for similar parts.

E. Ablations

Self-Supervision Ablations. In addition to the network described in Section 3 of the main paper, we also tried using a truncated SB-GCN (only its upwards pass) as the encoder network, using the same shape parameter input features. We evaluate both explicit surface and SDF accuracy in Table 1 and see that our architecture significantly outperforms SB-GCN for both measures.

Model	XYZ Error	SDF Error
Ours	**0.0256**	**0.0147**
SB-GCN	0.035	0.0214

Table 1. Self-Supervision ablations. XYZ Error is the average pointwise distance between predicted and sampled surface position. SDF Error is the average absolute difference between predicted and actual SDF value.

Segmentation Ablations. We tried three types of face-level prediction network using our self-supervised face embeddings to determine which was best. The first two try to directly classify faces from the embedding, one using a linear support vector machine (SVM) and the other using a multi-layer perceptron. These test linear and non-linear decision boundaries based on face-codes alone. The third is the message passing scheme described in Section 4, which tests if neighborhood context is necessary.

To evaluate how well each method worked across labeled dataset sizes, we trained each model repeatedly on all supported tasks at a variety of training set sizes using a scheme similar to our baseline comparisons. Table 2 records the average face segmentation accuracy across dataset sizes for both segmentation tasks. In most cases a non-linear decision boundary is more accurate than a linear one, and neighborhood information improves accuracy leading us to choose the message passing network as our method. For the Fusion 360 Segmentation task the simpler architectures performed very slightly better than the message passing network at low training set sizes, and for MFCAD SVM performed significantly better. We hypothesize that this is due to a combination of SVMs known ability to generalize well with few examples.

Classification Ablations. We also tested adding message passing layers prior to pooling for the classification network. As Table 3 shows, this additional complexity did not yield any improvement.

F. Effect of Rasterization Accuracy

Segmentation accuracy is correlated with the accuracy of our model's rasterization. Figure 6 shows how the likelihood of correctly classifying a face decreases with the reconstruction accuracy of that face. While this figure aggregates data from all training set sizes, the trend is the same across each of them, merely with different slopes. This suggests that downstream task performance could be substantially improved by improving the rasterization learning accuracy, allowing us to learn better representations of complex faces.

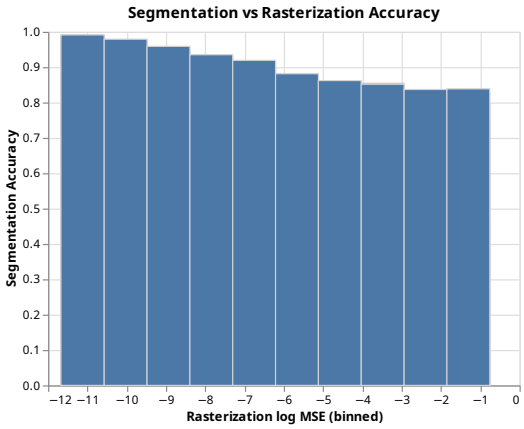

Figure 6. Segmentation accuracy compared to rasterization accuracy on the Fusion 360 segmentation task. The X-axis is the log of the per-face MSE of our rasterization, binned into 10 groups; the Y-axis is the fraction of faces segmented accurately within each bin. Data is aggregated across all segmentation training set sizes and seeds (the trend is similar across all sizes, with lower accuracies at lower training set sizes).

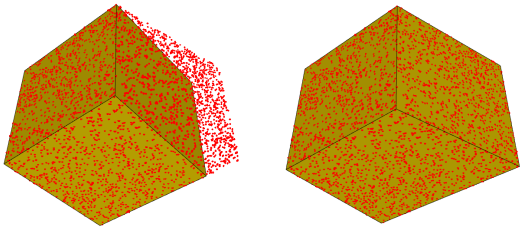

Figure 7. Shape matching with differentiable rasterization. The input cube (left) was optimized using SGD over its B-Rep shape parameters (right) to match a target point cloud (red).

G. Additional Examples

Here we present additional results on randomly sampled parts in rasterization and segmentation. Figure 8 shows reconstruction results on random parts from the Fusion 360 Gallery Segmentation test set. Figure 9 shows random part comparisons against baselines for construction based segmentation on the same dataset. Figure 10 shows additional manufacturing based segmentation results on random parts sampled from the MFCAD test set.

H. Gradient Based Optimization

Operating our self-supervision network as a rasterizer creates, in effect, a differentiable CAD renderer, we can use it for gradient-based optimization of B-Rep *shape parameters*. To demonstrate the potential of such an application, we prototyped a shape matching application, where an input B-Rep is optimized via stochastic gradient descent to match a target point cloud. Figure 7 shows the results of using this

Task / Model		Training Set Size					
Fusion 360 Segmentation	Accuracy@:	10	100	1000	10000	20000	23266
Self-Supervision + SVM		**0.66**	0.79	0.85	0.87	0.87	0.87
Self-Supervision + MLP		0.65	**0.80**	0.90	0.94	0.94	0.94
Self-Supervision + MP		0.65	0.79	**0.91**	**0.95**	**0.96**	**0.96**
MFCAD	Accuracy@:	10	100	1000	10000	13940	--
Self-Supervision + SVM		**0.40**	0.51	0.56	0.57	0.57	
Self-Supervision + MLP		0.36	0.60	0.86	0.93	0.93	
Self-Supervision + MP		0.35	**0.66**	**0.96**	**0.99**	**0.99**	

Table 2. Segmentation ablations. Reported face classification accuracy shows the mean of 10 runs at each dataset size with the train set subset at different random seeds (each model sees the same 10 random subsets). Models were selected by best validation loss on a random 20% validation split, except for the SVM models. Bold indicates the best accuracy at each train size for each task.

Task / Model		Training Set Fraction						
FabWave	Accuracy@:	1%	5%	10%	20%	50%	75%	100%
Self-Supervision + Pooling		.895	**.989**	.994	**.997**	**1.00**	**1.00**	**1.00**
Self-Supervision + MP + Pooling		**0.899**	**0.989**	**0.994**	0.997	**1.00**	**1.00**	**1.00**

Table 3. Classification ablations. Reported accuracy shows mean of 10 runs, similar to Table 2. Adding message passing prior to pooling does not confer an advantage, so we do not use it in our reported results.

optimization to angle the face of a cube. We find that this technique struggles on more complex shapes, which may overcome by improved rasterization performance.

References

[1] Open CASCADE Technology OCCT. http://www.opencascade.com/. Accessed: 2022-05-19. 2

[2] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 1

[3] Benjamin Jones, Dalton Hildreth, Duowen Chen, Ilya Baran, Vladimir G. Kim, and Adriana Schulz. Automate: A dataset and learning approach for automatic mating of cad assemblies. *ACM Transactions on Graphics*, 40(6), dec 2021. 1

[4] Joseph G. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. BRepNet: A topological message passing system for solid models. *arXiv:2104.00706 [cs]*, Apr. 2021. arXiv: 2104.00706. 1

[5] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[6] Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *CVPR*, 2019. 2

[7] Jiaxuan You, Rex Ying, and Jure Leskovec. Design space for graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 1

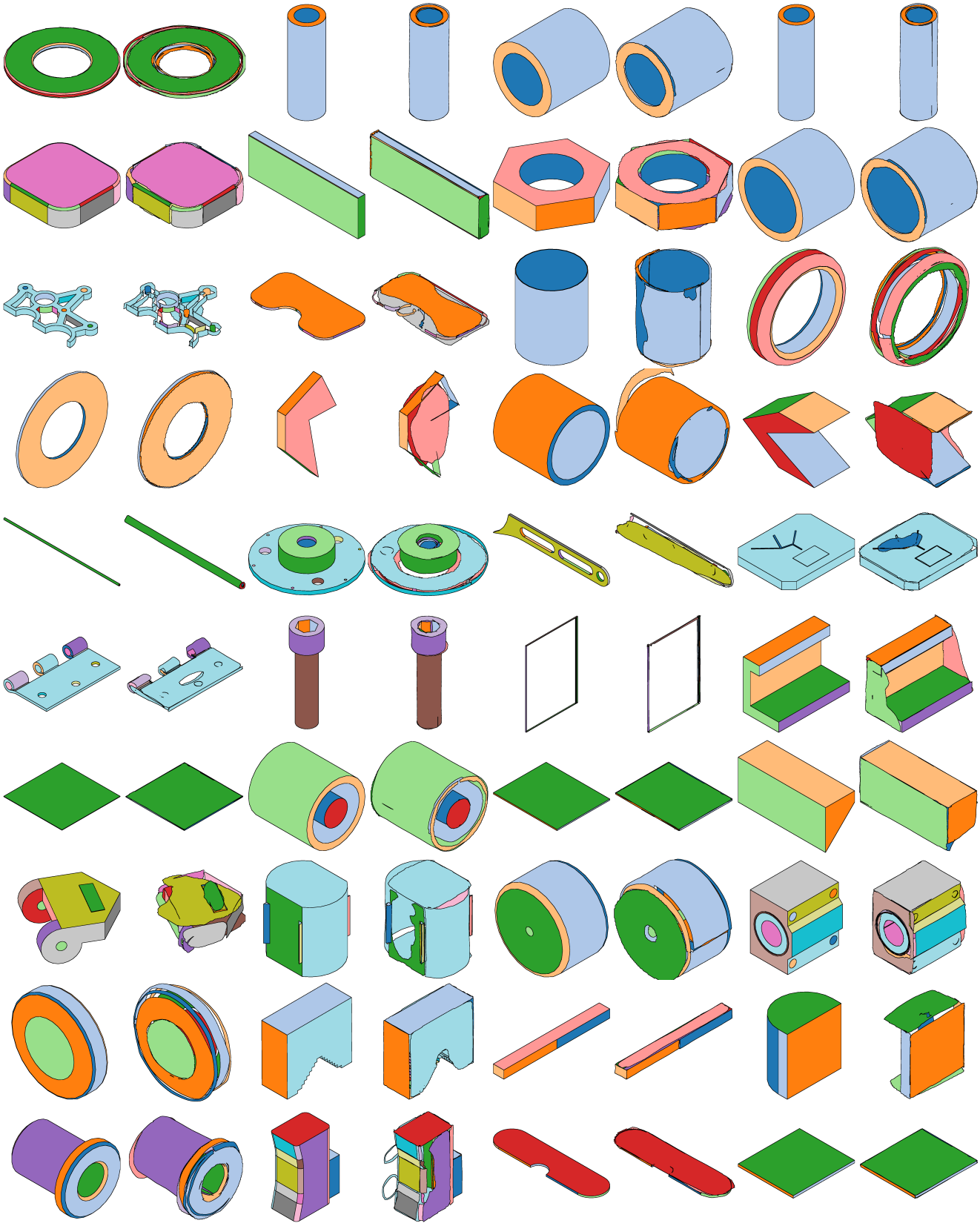

Figure 8. Additional reconstruction examples sampled randomly from the Fusion 360 Gallery Segmentation test set.

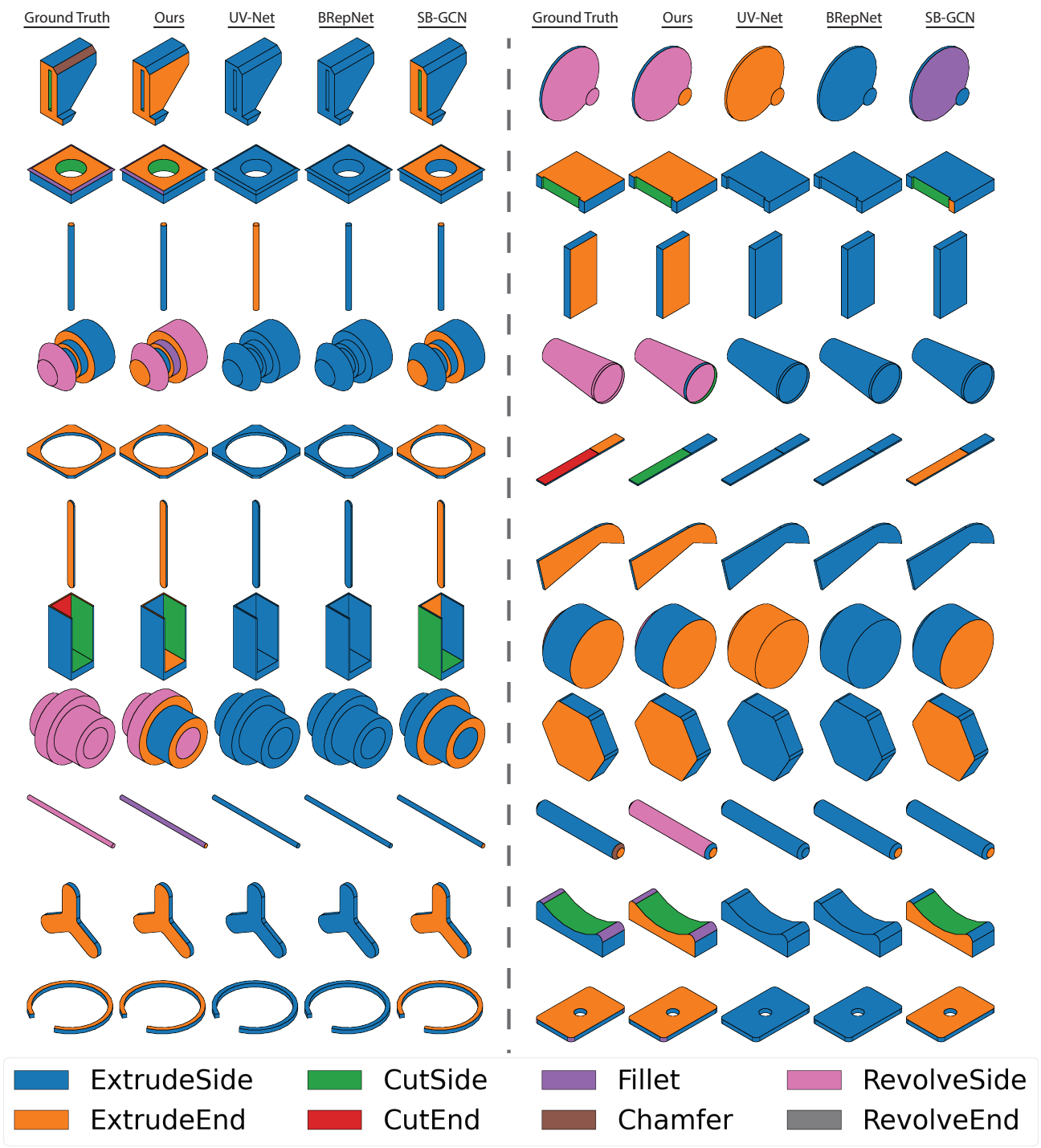

Figure 9. Additional construction based segmentation examples sampled randomly from the Fusion 360 Gallery Segmentation test set.

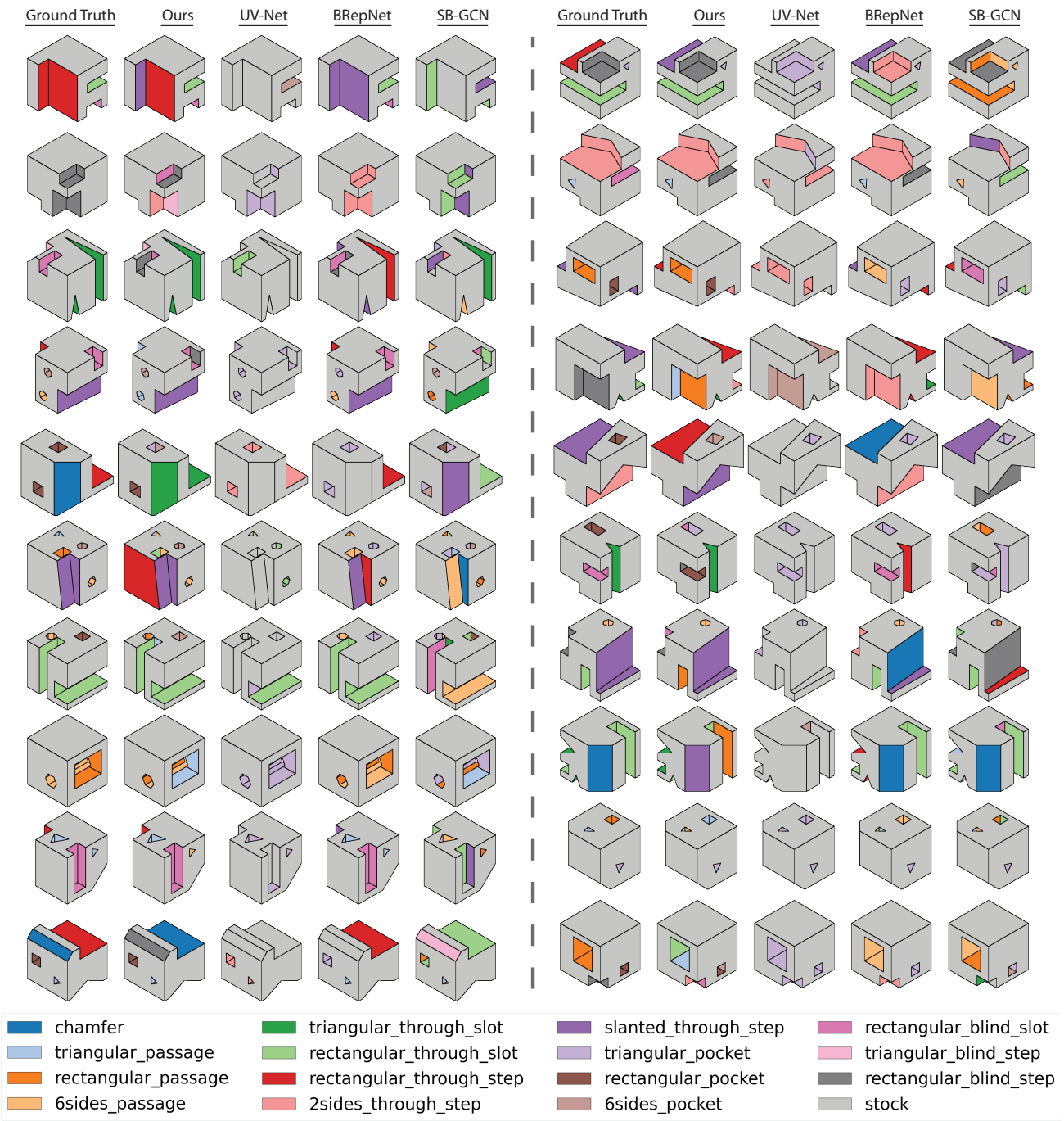

Figure 10. Additional manufacturing based segmentation examples sampled randomly from the MFCAD test set.