Supplementary Material: Bootstrap Your Own Correspondences

A. Implementation Details

RANSAC Baselines. We use the Open3D [5] RANSAC implementation and use the same parameters for all experiments, including our features. We run RANSAC for 10^5 iterations with estimates being scored on the number of inliers (within a threshold of 3 cm). It is worth emphasizing that this work is focused on learning better geometric features. As a result, our comparison here is against other features (*e.g.*, FPFH and FCGF), not against RANSAC. Since a better tuned RANSAC (*e.g.*, more iterations) would improve the performance of all methods, it does not affect any of our contributions.

Registration loss. Our registration loss is the weighted mean-squared error between the aligned correspondences. As we note in Sec. 3.1, the loss is back-propagated to the encoder through both the weights w *and* the transformation $\mathbf{T}$. Hence, it would be possible to train the models by applying losses on just the weights or just the transformation. In our experiments, we found that we got the best results by training the visual encoder using the weighted errors and training the geometric encoder without using the weights. Hence, the geometric registration loss is simply the mean-squared error between the aligned correspondences, without weighting. This only affects the loss calculation; we still use the weights when estimating the transformation for both feature modalities.

B. Qualitative Results

We present additional qualitative results in the supplemental material to provide a clearer picture of our model performance as well as its limitations. We visualize the extracted features using t-SNE [3], the resulting correspondences, and registration. We emphasize that the RGB images are *only* used for visualization; our model operates on the uncolored point cloud computed from depth images. We found that using color images to visualize the correspondences and to color the registered point cloud provided a clearer image of the model's performance.

t-SNE visualization. We visualize the features extracted from the point clouds by using t-SNE to map each vector from a 32-dimensional vector to a single dimension that is converted to a color using matplotlib's Spectral color map [2]. This is inspired by the visualization used by [1]. We use the features from both input pairs when calculating t-SNE. This allows similar features between both images to be mapped to similar colors.

Feature Extraction. As shown in Figure B.1, our model extracts features that appear to consistently map the same parts of the scene to similar features. This allows the model to extract correct correspondences and accurately register the point clouds. We found it interesting that the t-SNE mapping of the features shows discrete jumps around object boundaries. It would be interesting to explore the efficacy of our training pipeline for learning tasks like 3D object detection or semantic segmentation. While PointContrast [4] has explored the use of correspondence losses as a pretext task for pretraining networks with 3D segmentation and detection as downstream tasks. It would be interesting to see if a simple linear probe could allow us to learn accurate segmentation on top of our pretrained features, as well as how well this performs compared to a model trained using ground-truth correspondences.

Correspondence estimation. We also find that our model is able to extract accurate correspondences for different scenes and different degrees of overlap. As a result, our model is able to accurately register point clouds under a variety of setups. Furthermore, the density of extracted correspondences allows us to accurately register the object even if some of the correspondences are incorrect.

Failure modes. Finally, we also observe some failure modes in our model that are shown in the bottom 3 rows. In the seventh row, the model's correspondences appear to be mapping points on the sink's surface across the images, resulting in the registration being slightly off. We observe this pattern for images with large, fairly flat surfaces. This can be seen to a larger extent in the eighth row where the model extracts correspondences between random points on

the carpet, resulting in very poor registration performance. Finally, the bottom row shows a case where the incorrect depth estimation results in erroneous registration. In this case, the input incorrectly specified the open window as a flat surface, resulting in a large number of incorrect correspondences.

References

[1] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *ICCV*, 2019. 1

[2] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 1

[3] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008. 1

[4] Saining Xie, Jiatao Gu, Demi Guo, Charles R. Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *ECCV*, 2020. 1

[5] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv*, 2018. 1

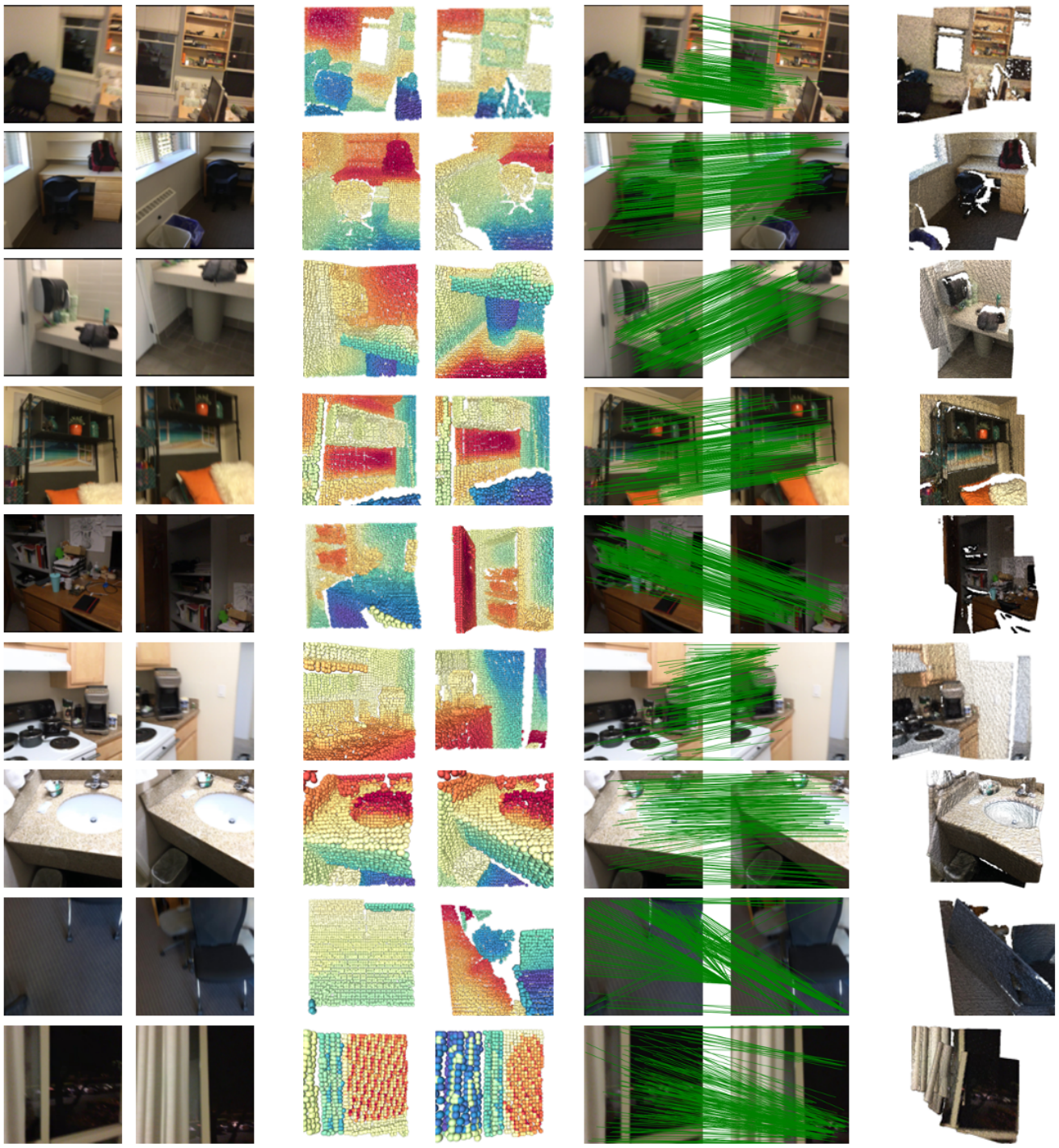

Figure B.1. Our model can accurately estimate correspondences and register point clouds for a variety of indoor scenes and differing degrees of overlap between point clouds. This is enabled by learning good point-level features as shown by the t-SNE visualization in the third and fourth columns. The distinctiveness of the learned features enables accurate correspondence estimation and registration. We observe several failure modes that can be caused by relatively flat geometry (rows 7 and 8) or incorrect depth input (row 9).