Single-Image Piece-wise Planar 3D Reconstruction via Associative Embedding

Machine Learning for 3D Geometry - Technische Universität München

Sonia Castro*, Laura Sáez*, Hassan Sarwat*, Dmitry Fadeev

Abstract

This document reflects the motivation, methodology and results of a project in machine learning for 3D geometry. In particular, it focuses on piece-wise planar reconstruction of a 3D scene from a single RGB image, building upon the work of Yo et al. *[13] by introducing several modifications with the intention of the study and potential improvement of their results. Our code is available at* `https://github.com/Hassan-Sarwat/PlanarReconstruction_ML3D`

1. Motivation

Humans effortlessly interpret a room by recognizing dominant planes and surfaces, yielding a simple, high-level understanding of the scene. Piece-wise planar reconstruction translates the concept to computer vision, by inferring a set of plane parameters and assigning a plane ID to each pixel [8]. As highlighted by previous works, this approach offers multiple advantages, from compactness and simplicity of decomposition, modeling and rendering, to movement and object placing in robotics and augmented reality [8]. Moreover, it overcomes the challenges of sharp-angled, poorly-textured surfaces of man-made structures, which create ambiguities and defeat smoothness assumptions [3, 4]. The advances in deep learning are allowing state-of-the-art results such as the ones presented in Yu *et al.*, our starting point.

2. Introduction

In this project, we intend to extend the work presented in Yu *et al.*, which aims to segment plane instances and simultaneously recover their 3D plane parameters from an RGB image. The main contribution of this paper is its novel two-stage approach based on associative embedding. Opposed to prior works, this methodology is bottom-up and uses CNNs, overcoming the drawbacks of both top-down CNN approaches (fixed number of planes) and bottom-up geometric ones (challenging primitive detection, restricted applications, time-consuming optimization).

[0]The authors with * had an equal contribution.

2.1. Baseline Approach

As introduced, Yu *et al.* present a multi-branch, end-to-end model, able to infer an arbitrary number of planes. Their proposed architecture consists of 2 stages and 3 branches. Firstly, a ResNet encoder embeds each pixel through associative embedding, providing shared features for all branches. The first stage predicts, on the one hand, planar/non-planar masks to only consider the former, and, on the other, the pixel-wise embeddings meant to pull same-instance pixels together while pushing the rest. These results are merged and grouped into plane instances via an efficient mean shift clustering algorithm. The second stage, focuses on plane parameter estimation, at both pixel and plane level, weighting the latter by the segmentation probabilities of the previous stage. Geometry consistency is enforced by evaluating the inferred depth map from the predicted parameters, finalizing the end-to-end learning pipeline. A detailed figure of this architecture from the original paper can be found in Appendix A (Fig. 2). For further details, we encourage referring to the original paper.

2.2. Proposed Modifications

Our contribution aims to build upon their methodology by introducing several modifications to their work and evaluating their performance. In particular, we aim at attaining or improving the presented results by focusing on the encoding and embedding of the pixels, as well as leveraging the data with semantic information. For the encoder model that provides the shared features used afterward, we propose a Dense Prediction Transformer (DPT) instead of the original ResNet-101-FPN [5]. Regarding the evaluation of these embeddings, our proposal is a data-efficient version of the supervised contrastive loss. Furthermore, we suggest adding a fourth branch to the model, that uses the semantic information of the scene in order to leverage the current information and try to improve the context understanding. In the following section, we expose the technical aspects of each modification, motivated and justified by our research and the encountered challenges. We also provide a brief explanation of the dataset requirements for this work. Our proposed architecture can be seen in Fig. 3.

3. Method

3.1. Dense Prediction Transformer

As mentioned in [6] downsampling using CNNs suffers from losing feature resolution and granularity in the deeper stages of the model. For this purpose, we decided to use a DPT from the Hugging Face Library pretrained on ADE20K Dataset [11] on the task of semantic segmentation.

Unlike Resnet, which uses convolutions to downsample an image, a Dense Prediction Transformer embeds image patches or features into embeddings using bag of word image representation [9], then the transformers [10] transform the embeddings using multi-headed self-attention blocks (MHSA) [10]. This means that the number of embeddings is maintained throughout the entire transformation so no information is lost, and MHSA is inherently a global operator, so all embeddings have access to each other throughout the entire network, unlike CNNs which progressively increase the receptive field.

We faced 2 problems when switching the architecture, the first was that DPT only had the encoder pretrained, which outputted embeddings instead of image like feature representations. To solve this we also implemented the convolutional decoder part of the DPT, however we couldn't find a pretrained version of it so we had to train it from scratch. The second problem was image size, as DPT only accepted square images whereas our model was working on 192 × 256 sized images, so we applied a `DPTImageProcessor` from Hugging Face Library on the input images also pretrained on the ADE20K Dataset. After that, we implemented our own head to fit the resulting image or feature representations into the expected dimensions for the next model stage.

This head was inspired by the Semantic Segmentation head in the DPT For Semantic Segmentation (`DPTSemanticSegmentationHead` function), although we don't use the dropout, and we follow it with a convolutional layer reducing the number of channels, followed by an average pooling layer to downsize the height dimension of the features to 192 from 256, and a final ReLU activation function before passing it to the next model step.

The code of our dpt model can be seen in `baseline_scene.py`, the architecture can be viewed in the initialization of the `Baseline` class and the forward pass can be seen in `dpt_backbone` function.

3.2. Contrastive Loss

The original proposal uses a discriminative loss function presented in [1], a pull-push loss based on the Euclidean distance of pixels to the instance centers and the use of margins. Ours is a data-efficient version of the supervised contrastive loss, later presented in [7]. It is defined as follows for a single pixel i, where z_j denotes the embedding representation of pixel j and $P(i)$ is the set of positive pixels out of all the ones being considered $A(i)$:

$$\mathcal{L}_i = \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p)}{\sum_{a \in A(i)} \exp(z_i \cdot z_a)} \tag{1}$$

Nonetheless, performing pairwise comparisons for each pixel in our images of size 192 × 256 and operating with the resulting matrix is not feasible. Instead of this quadratic algorithm w.r.t. the number of points N, our approach uses the centers and a fixed number of anchors m per plane p, with $mp \ll N$, attaining a much tractable complexity of $O(N \cdot p(m+1)) \ll O(N^2)$. We considered three variations of this loss. The default is `contrastive_loss`, which only considers the centers of each instance ($m = 0$) as anchors, so that $|A(i)| = p$ and $|P(i)| = 1$. The `contrastive_loss_anchors` also takes the center of m subsamples of each instance, of size defined by a fixed proportion. This way, we have $|A(i)| = (m+1) \cdot p$ and $|P(i)| = m + 1$. Moreover, we also considered using only extra negative anchors in `contrastive_loss_anchors_neg`, a variant of the former, but only considering the true center as a positive. Nonetheless, this has been empirically discouraged in [7] itself, through an ablation study.

The code can be found in `contrastive.py` in our repository. We compared these versions in a smaller setup, and performed fine-tuning on m in Appendix C.

3.3. Semantic Branch

The original paper highlighted potential future directions, one of which involves the integration of semantic information. In our efforts to address this, we incorporated a semantic decoder as a convolutional layer, positioned above the backbone features. This decoder has 41 output channels, 40 for the possible classes in the NYUv2 dataset and an additional channel dedicated to the "unknown" label. The training process utilized the `CrossEntropyLoss`.

Our objective was to incorporate this information into the generation of plane instance segmentation. To achieve this, we concatenated the plane embeddings (comprising 2 channels) with the output derived from the semantic decoder. Subsequently, we convoluted these concatenated features to obtain the 2-channel input necessary for the implemented mean shift algorithm. This aims to enhance the segmentation process by leveraging both plane embeddings and semantic information.

As an alternative approach, we explored the addition of an extra convolutional layer, transitioning from the 41 channels of the semantic decoder to 2 channels before the concatenation with the plane embeddings.

3.4. Dataset

Our paper utilizes preprocessed data obtained from PlaneNet [8], 51,000 ground-truth piecewise planar depth maps sourced from ScanNet [2]. To fill depth map gaps, planes are fitted to a mesh and projected back to frames using semantic annotations. Refer to the original paper for more details.

Using ScanNet++ [12] with the provided Renderer revealed disparities in files and directory structure. Consulting ScanNet_v2, led to the use of SensReader. Renderer and SensReader codes were updated for compatibility with the latest package versions, they can be found in our github repository. The process of generating the final `tf_records` was unavailable and the output of the renderer lacks adequate documentation. To address this limitation, we relied on our model's input requirements (Appendix B.1) and created a script to extract them, incorporating this ScanNetScene to generate the segmentation ground truth from the renderer output. However, certain indices in the segmentation ground truth were beyond the defined bounds. Unfortunately, the source of these errors couldn't be identified, so we resorted to the original dataset.

4. Results

We run our experiments with a subsample of 10,000 images for training w.l.o.g. in our results (Appendix B.2). The training is done for 5 epochs to allow more experiments. Configuration remains unchanged w.r.t. [13], except for the DPT, where a lower learning rate and weight decay were used for convergence. The details are in `config.yaml`.

Our main result is shown in Fig. 1a, which displays the pixel and plane recall curves of the baseline and the best case of each presented modification. The quantitative evaluation is not satisfactory, as the behavior of both the semantic branch and the contrastive loss is significantly undesired. The combination of modifications is not considered due to its detrimental effects.

Fig. 1b is its qualitative counterpart, showing the plane segmentation prediction for validation sample `2.npz` (arbitrary choice). The pictures ratify the numerical concerns. It appears that our embedding loss is incapable of grouping the embeddings in planes, but rather creates serpentine groups that slightly resemble the overall image structure as a whole. Even more extreme is the case of the semantic branch, which seems to be predicting a single plane for all pixels in the image. Several attempts with other files confirmed this to be a general pattern rather than a failure case. For this reason, we performed a throughout assessment of our results in more detail.

4.1. Dense Prediction Transformer

Although we achieved slightly lower recall values than for the baseline, the encoder results align with our expectations, due to the challenges exposed in Sec. 3.1. The closeness to the baseline is enough to consider the method effective, taking into account the preconditions.

4.2. Contrastive loss

We tested the training performance of our 3 variants, as well as the effect of the m parameter in Appendix C. Although the one with no anchors shows a slight decrease, the overall takeaway is that our approach is incapable of learning separable embeddings aligning with the plane instances. Nothing indicates that the anchors' addition offers any improvement, even after normalization to mitigate the proportional loss value increase w.r.t. m. In fact, Fig. 5 shows the best approach to be the default one.

Regarding the number of anchors m, once a significant number of them are added (i.e. 50), neither increasing (100) or decreasing (20) appears to have any effect. The training curves in Fig. 6 show almost the exact same flat pattern but at different scales, corroborating the superiority of the default ($m = 0$) loss, as training time has also been shown to increase with m.

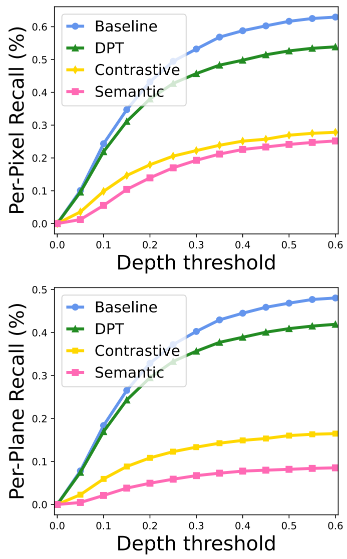
Finally, taking advantage of the embedding size being 2, we plot them in Fig. 7 for two arbitrary validation samples `2.npz` and `41.npz`, the first being the one from the results. Nonetheless, this turned out to be less interpretable and clear than we expected.

4.3. Semantic Branch

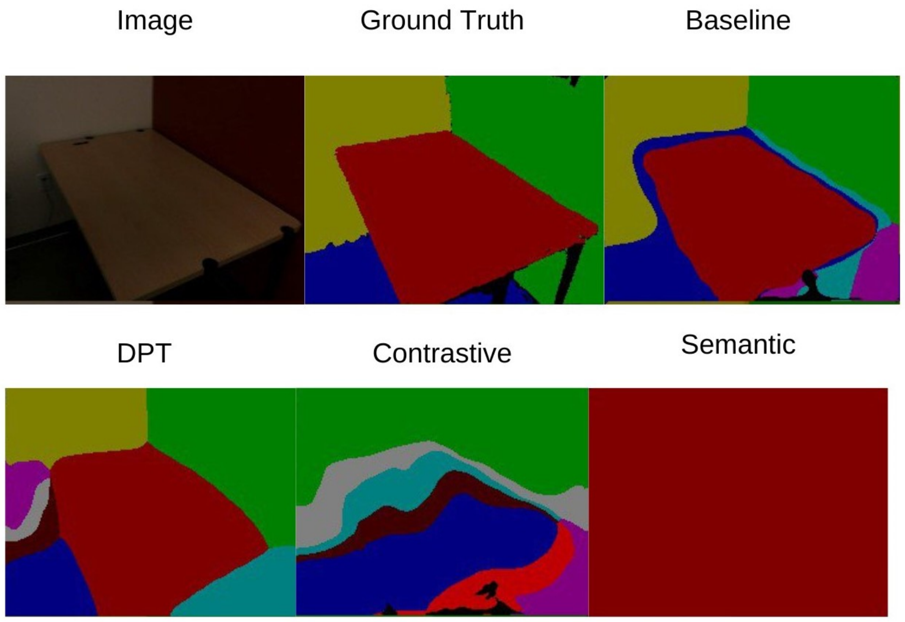
The main goal of this addition was to improve the downstream task (plane prediction) performance of the model by adding semantic information to each pixel. Nonetheless, our results show that instead of an improvement, the model completely collapses and resorts to a single-plane prediction for every pixel in an image.

The main surprise comes from the quantitative and qualitative evaluation of the semantic branch itself. Both of the approaches described in Sec. 3.3 show a learning capability in their decreasing training curves despite their poor recalls, in which it is demonstrated that the addition of an extra layer is detrimental (see Fig. 8).

Moreover, although far from perfect, the segmentation results in Fig. 9 are sensible and satisfactory. Opposed to the contrastive loss, the problem is not in the modification itself. Even though these are training results, overfitting is discarded because the same single-prediction response is already observed during training.



(a) Pixel (up) and plane (down) recall curves of the best variant of each proposed modification.



(b) Qualitative validation results of each of our proposed modifications compared to the ground truth and baseline for the plane segmentation (`pred_seg` and `gt_seg_image`). First subimage shows the original image, which corresponds to `2.npz` in the validation set.

Figure 1. Main quantitative and qualitative results of the project, using the same performance metrics as in the original paper.

5. Conclusions

We now offer hypotheses for the reasons behind the failures, as well as changes that could potentially mitigate them. The paper's reproducibility and clarity is also assessed, as it supposed one of our greatest challenges.

The DPT is our most successful modification, achieving values close to the baseline with a difference explained by having to resize the non-square images to fit the input and our convolutional decoder not being pretrained, as well as the adjustments to fit with the rest of the architecture. Possible improvements would include further training (both in epochs and samples) of the decoder, and editing the entire model to fit 256×256 sized images.

The contrastive loss, despite learning some structure (some groups combined resemble the image rather than being completely random) does not perform adequately. The sub-center strategy is not successful either, likely because taking subsamples' mean adds no information, yielding almost identical anchors and explaining Fig. 6. Perhaps a better approach would be random sampling, although computationally more expensive. Nonetheless, the underlying issue appears to go deeper. We do not discard a conceptual from-the-start misuse, or that an embedding dimension of 2 might be enough for L2 distances but not to capture similarities with our loss, although Fig. 7 was not of any help to confirm or dismiss this.

Finally, our attempt to leverage the encoded features with semantic information proved to be detrimental, rather than beneficial. On an abstract level, a single plane can have multiple objects on it, and two different planes can be of the same class, which is enough to explain why it could fail. On a technical scale, although we considered other combination strategies (i.e. concatenation instead of convolution), `bin_mean_shift.py`'s hard adaptability constricted its input dimension. Regardless, the ground truth and its distribution shows the problem to be quite ill-constrained: a significant proportion of the pixels are labelled as unknown or others (encouraging ambiguity and similarities between different plane instances), as shown in Fig. 9 and Fig. 10.

Finally, we want to remark the high complexity behind adapting Yu *et al.*'s work to new data. PlaneNet's preprocessing is barely motivated or referenced, the required steps to achieve an input are completely absent, and even [8] profoundly lacks documentation. The provided set of `.tfrecords` has cryptic origins without any further mention. As a whole, it is exceptionally hard to further work on their setup. Despite our many efforts and positive results exposed in Sec. 3.4, the overall outcome was not functional.

All in all, regardless the actual results, we performed an extensive analysis on our project paper, applied the proposed modifications and assessed their results, considering the reason behind their shortcomings and possible future work to overcome them.

References

[1] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *CoRR*, abs/1708.02551, 2017. 2

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 3

[3] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Manhattan-world stereo. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1422–1429, 2009. 1

[4] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1418–1425, 2010. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 2

[7] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *CoRR*, abs/2004.11362, 2020. 2

[8] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single RGB image. *CoRR*, abs/1804.06278, 2018. 1, 3, 4

[9] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):591–606, 2009. 2

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. 2

[11] Weihao Xia, Zhanglin Cheng, and Yujiu Yang. Sr-net: Cooperative image segmentation and restoration in adverse environmental conditions. *CoRR*, abs/1911.00679, 2019. 2

[12] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023. 3

[13] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. *CoRR*, abs/1902.09777, 2019. 1, 3

A. Model architectures

In this section we provide the figures showcasing both the original architecture in Yu *et al*. and a modified version of the same image including our modifications, as explained in Sec. 2.

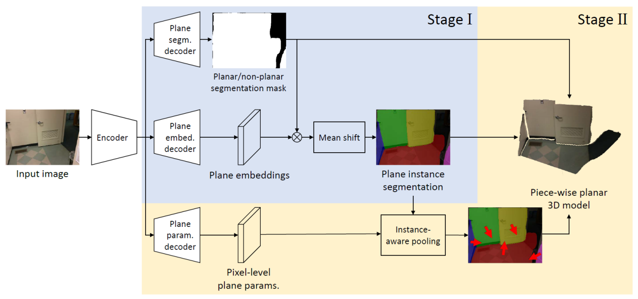


Figure 2. *Caption from the original figure.* Network architecture. In the first stage, the network takes a single RGB image as input, and predicts a planar/nonplanar segmentation mask and pixel-level embeddings. Then, an efficient mean shift clustering algorithm is applied to generate plane instances. In the second stage, we estimate parameter of each plane by considering both pixel-level and instance-level geometric consistencies

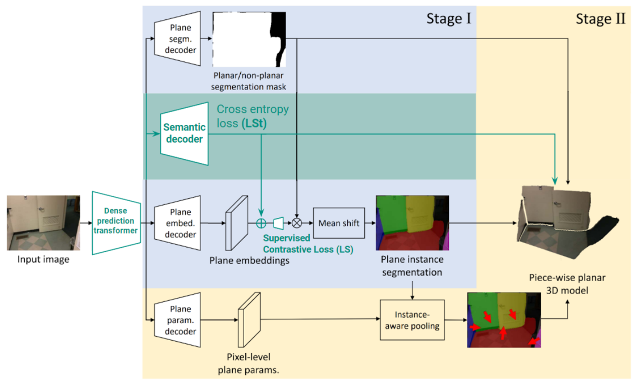

Stage I
Stage II
Plane segm. decoder
Planar/non-planar segmentation mask
Cross entropy loss (LSt)
Semantic decoder
Input image
Dense prediction transformer
Plane embed. decoder
Supervised Contrastive Loss (LS)
Plane embeddings
Mean shift
Plane instance segmentation
Piece-wise planar 3D model
Plane param. decoder
Pixel-level plane params.
Instance-aware pooling

Figure 3. Our modifications, as explained in Sec. 2.2. The parts we changed or added are marked in turquoise. Special emphasis is to be put in the semantic branch, where we concatenate the embeddings with the semantic information and process them convolutionally to yield the total combination.

B. Dataset

B.1. Input format

The required input to the model is a npz file containing: An image with shape (192, 256, 3), the depth ground truth shape (192, 256, 1), the semantic ground truth shape (192, 256), the number of planes in that frame shape (1,), the plane parameters shape (K, 3) where K is the number of planes in the frame and the ground truth segmentation shape (192, 256, 1).

B.2. Sampling

In order to ease computation and data downloading and transferring, the project experiments are done using a subsample of the original processed ScanNet dataset. Therefore, from the previously mentioned 51000 samples, out of which 50000 were used for training, we take a subsample of 10000. We ensure the validity of our subsample by studying the distributions of both the whole dataset and our subsample, which is taken randomly. In particular, Fig. 4 compares the distribution of the number of planes and semantic classes in every image, as well as the distribution of semantic classes pixel-wise across the whole sets. An additional important check is the proportion of planar and non-planar pixels across both sets, which is of 0.791 and 0.792. With all these, we can assume equal distribution and no loss of generality.

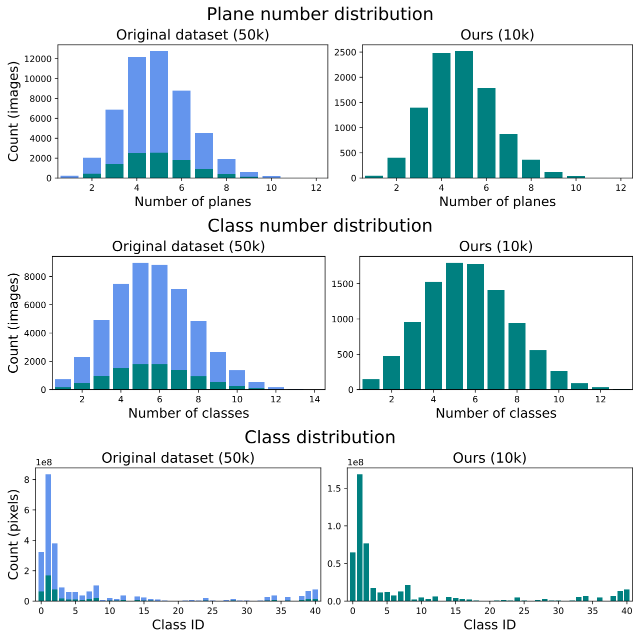


Figure 4. Comparison between the original dataset (left, blue) and our subsample (right, turquoise) distributions. The subsample is also shown overlapped in the left side plots. Both distributions being equal implies that the presented results can be extrapolated to the whole dataset w.l.o.g.

C. Contrastive loss additional results

This section contains the additional experiments we did in order to choose the best contrastive loss variant and evaluate the effect of the number of subcenter per-plane anchors m. Given the both poor and unexpected results that this modification has shown, specially when looking at the segmentation predictions, we also performed a study on the embeddings of a couple of images after training.

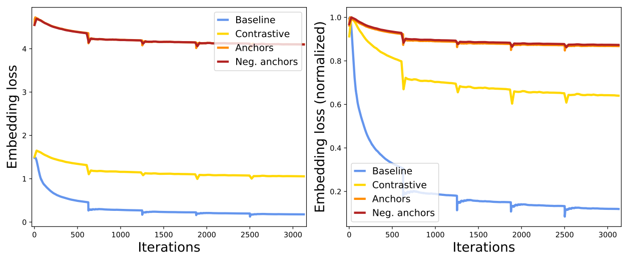


Figure 5. Training curves of the plane embedding branch, comparing the loss of our three proposed losses against the discriminative baseline. Losses are shown in both absolute (left) and normalized (right) values.

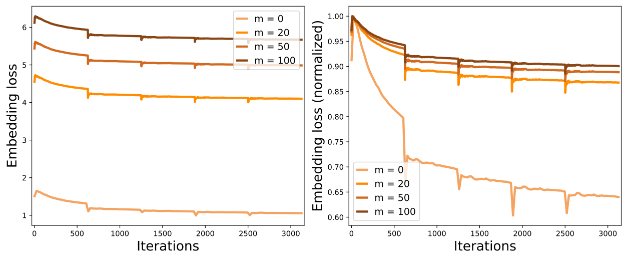


Figure 6. Training curves of `contrastive_loss_anchors`, comparing the performance w.r.t. the value of m. Losses are shown in both absolute (left) and normalized (right) values.

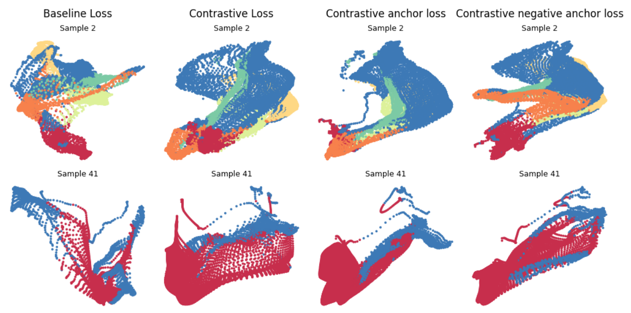


Figure 7. Embedding visualization of evaluation samples `2.npz` and `41.npz` for the models trained with each embedding loss. The coloring represents ground truth instances.

D. Semantic branch additional results

This section covers the additional experiments and visualizations we performed for the semantic branch, showing both qualitative results during training (which seem sensible and functional), and quantitative results in training and testing. Although the branch itself is capable of performing its local task, it not only does not help for the overall one, but obstructs it.

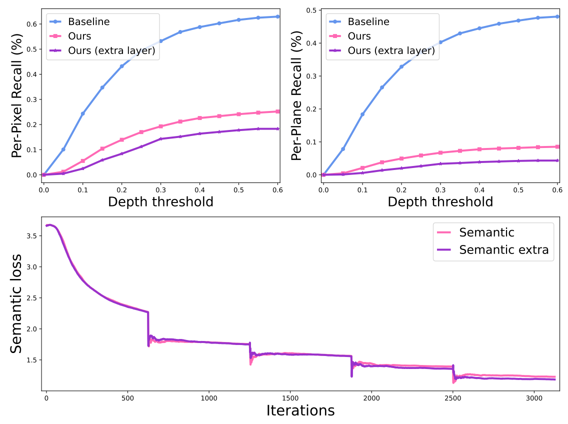


Figure 8. Semantic branch quantitative results including the extra layer and without it. Both the evaluation recall curves (top) and the training loss curves (bottom) are displayed.

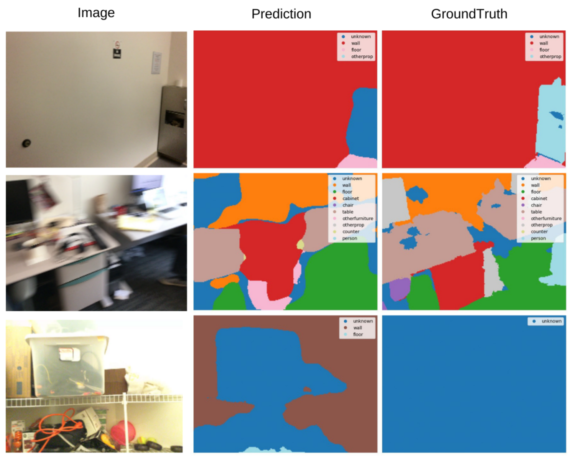


Figure 9. Predicted semantic labels from the semantic decoder and their corresponding ground truths. These are training unknown arbitrary samples.

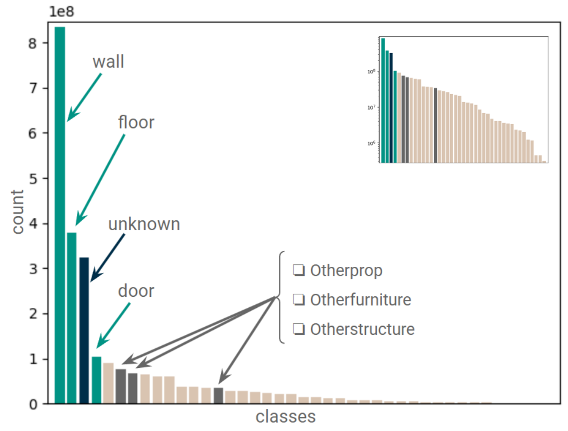


Figure 10. *Figure from the presentation.* Class distribution taken per-pixel from the whole 50k dataset. Upper right shows log-scale.