Exploration of Structural-Textural Line Segment Classification

Yifan Yu [1] **Shinjeong Kim** [1] **Alexis Tabin** [1] **Marco Voegeli** [1]

Abstract

Lines reflect the high-level structural layout of 3D scenes. Existing line detection methods on 2D images lack information on whether the 3D correspondence of a line segment is structural or textural, which could be useful for downstream tasks like 3D reconstruction. We present the very first attempts to classify line segments by this criteria. By predicting pixel-wise surface normal differences using neural networks, our method provides reasonable classification while being detector-agnostic. The classification helps to create a cleaner reconstruction of 3D scenes. We also explore end-to-end models to perform detection and classification simultaneously.

1. Introduction

In computer vision, line segments are useful in a wide range of tasks such as SLAM (Zuo et al., 2017; Pumarola et al., 2017), pose estimation (Xu et al., 2017; Gao et al., 2021), and 3D reconstruction (Hofer et al., 2017; Zhou et al., 2019). Compared to feature points, line segments provide more information on the high-level structural layout of 3D scenes and are more robust to viewpoint and illumination changes.

Conceptually, structural lines (e.g. edges of walls, ceilings, floor) provide more structural cues than repetitive textural lines (e.g. grid lines of floor tiles) in 3D scenes, especially in man-made architectures. Therefore it could be beneficial for downstream tasks, such as reconstruction, scene understanding, and plane detection if such classification of line segments could be recovered. Existing line segment detectors work well in recovering 2D line segments from images. They, however, do not provide information on whether a line segment is more structural or textural.

In this report, we present the first attempts at this novel task: classifying line segments detected in 2D images based on whether they correspond more to structural or textural lines in original 3D scenes. Our proposed method utilizes the fact that structures could be reflected from changes in surface normal directions: we train proven deep learning models in computer vision to estimate pixel-wise value reflecting the change in surface normals in the vicinity, which is then used to classify the line segments in a detector-agnostic manner through a filtering step. Ground truth surface normals from synthetic dataset Hypersim (Roberts et al., 2021) are processed to generate the training data. Experiments demonstrate our method is able to reasonably separate the structural line segments from textural ones in images, works better than filtering using predictions of the state-of-the-art surface normal prediction model, and provides useful information for downstream line-based vision tasks like 3D line reconstruction. Enabled by our data processing method, we are also able to train a complex end-to-end model adapted to our task to detect line segments and their structural-textural classification simultaneously with promising results.

2. Related Work

Line Segment Detection. Conventional line segment detectors such as LSD (Von Gioi et al., 2008) and EDLines (Akinlar & Topal, 2011) rely on grouping image gradients. Deep learning-based detectors offer better results in particular for specialized tasks like wireframe parsing (Huang et al., 2018; Zhang et al., 2019; Xue et al., 2020; Huang et al., 2020; Pautrat et al., 2021; Yoon & Kim, 2021). Gradient-based detectors produce more segments and with higher pixel accuracy by nature, therefore are good for tasks that benefit from more correspondences and higher accuracy like visual localization. Deep learning-based methods like SOLD2 (Pautrat et al., 2021) generally focus more on segments that are semantically more structural, and more suited for tasks like 3D line reconstruction. Our method tries to classify the segments disregarding the detectors in use, to provide additional information for downstream tasks.

Depth and Surface Normal Estimation. Existing works have achieved promising results on surface normal estimations via monocular depth estimation (Ranftl et al., 2020; 2021), or directly predicting surface normals (Chen et al., 2017; Eftekhar et al., 2021). However, the state-of-the-art surface normal estimation model Omnidata (Eftekhar et al., 2021; Kar et al., 2022) still shows significant angle errors and are not accurate enough to classify line segments using differences in the predicted normals as we will show later.

[1]ETH Zürich, Zürich, Switzerland.

Deep Learning, ETH Zürich, 2022

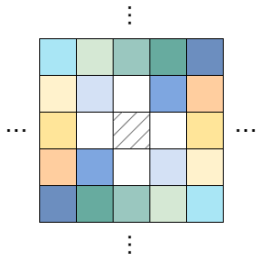


Figure 1. Illustration of surface normal difference calculation. The cosine value of angles between surface normal directions at each pair of pixels symmetrical across the center pixel (marked with same color) are calculated using the dot product. Then the mean of the cosines is taken as the normal difference value at this pixel. In practice, we consider a 5x5 neighborhood and take pairs of pixels with a Manhattan distance larger or equal to 2 to the center pixel.

3. Method

In this section, we first introduce how the data is processed to generate pixel-wise values evaluating the surface normal difference within its neighborhood and how the values are able to reasonably classify the line segments detected in the images into structural and textural line segments using a filtering step. Then, we present the models and the training procedure for estimating per-pixel surface normal differences using the processed data, which allows us to separate detection and classification in a detector-agnostic manner. Finally, we also present our exploration of using end-to-end model to detect and classify the line segments (i.e. producing line segments and their structural/textural labels) simultaneously.

3.1. Data Processing

In 3D scenes, an intuitive characteristic of structural lines is lines where the surface normal changes significantly on the two sides. The most common cases are the ceiling and floor lines where the normal directions form a right angle, as well as edges between walls, that exist ubiquitously in man-made structures. With this insight, we evaluate the extent to which the surface normal directions change within a small neighborhood around each pixel and use this value on pixels passed by each line segment to estimate whether the segment is more structural or textural.

For this purpose, we use the photorealistic synthetic dataset Hypersim (Roberts et al., 2021) because it provides accurate ground truth surface normals. Due to time and resource limits, a subset of the entire Hypersim dataset is used in the experiments which contains roughly around 5000 images from 29 scenes from `ai_001_001` to `ai_003_010`, among which 3 scenes (around 700 images) are kept for validation and test and others for training.

Computing Normal Differences: As illustrated in Fig. 1, we evaluate the change of surface normal directions around a pixel by computing the cosine of the angle between normal directions at each pair of pixels symmetrical across the center pixel, and then take the mean value of the cosines to quantify the difference of surface normals around this point (referred to as normal difference at this pixel hereafter). In practice, the pairs of pixels are taken within a 5x5 neighborhood and with a Manhattan distance larger or equal to 2 to the center pixel, as we empirically observed that the immediate neighboring pixels often do not reflect the actual change of normal directions and doing so provide more reasonable classification visually. The resulting pixel-wise values in $[-1, 1]$ characterize whether the pixel is likely on a sharp structure like an edge or a corner (small values, normally close to 0) or on a smooth area like a plane (close to 1). The generated values are used to train our per-pixel normal difference estimation model as pseudo-GTs.

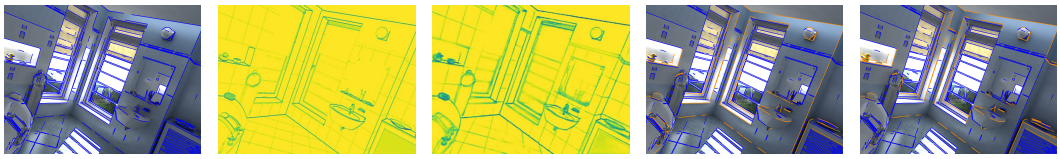
Classification by Filtering: With the per-pixel normal differences calculated from ground truth (or predicted from a deep learning model), we perform a filtering step on the line segments detected by some line segment detector: compare the median value of normal differences on pixels passed by each line segment to a threshold (cosine of a threshold angle), and label the line as structural if smaller than the threshold, and textural otherwise. The median value is best for this filtering because it's robust to outliers. The pixels passed by a line segment can be obtained using Bresenham's line algorithm (Bresenham, 1965). As Fig. 2(d) shows, this filtering is able to provide reasonable classification on detections of the LSD detector (Von Gioi et al., 2008).

3.2. Detector-agnostic Classification

With the described data processing and filtering method, we are able to separate line segment detection from the structural-textural classification and perform the classification in a detector-agnostic manner, using the surface normal differences estimated by a computer vision deep learning model. The problem is therefore modeled as a per-pixel regression of surface normal differences.

Due to the similarity in per-pixel prediction to object detection or semantic segmentation, we adopted widely-used architectures for these tasks, namely the Feature Pyramid Network (FPN) (Lin et al., 2017) and U-Net (Ronneberger et al., 2015). These CNN-based model architectures encourage the model to learn from short and long-range structures forming the layout of the scene of our interest. Similar to the FPN paper, we adopted the ResNet (He et al., 2016) backbone but with a smaller size variant (ResNet34).

To make these per-pixel classification models work for our regression problem, the last layer and loss function need to be changed. Empirically, we found the binary cross-entropy loss applied on sigmoid activation is able to classify the lines



(a) (b) (c) (d) (e)

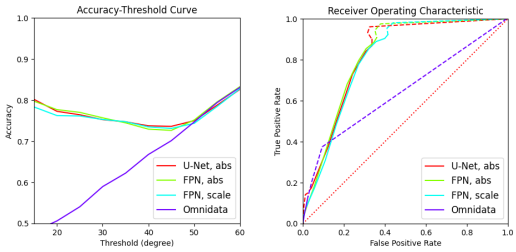
Figure 2. Visualization of line segment detections and classification results. (a) LSD detections; (b) Generated pseudo-GT normal differences; (c) Prediction from the model; (d) Classification using generated pseudo-GTs; (e) Classification using model predictions.

most accurately. However, a mapping from the generated pseudo-GT normal differences from $[-1, 1]$ to $[0, 1]$ (since they are cosine values) during training, or a mapping from the network output in $[0, 1]$ to $[-1, 1]$ for inferencing is needed. Among several alternatives, the mapping from pseudo-GT range ($[-1, 1]$) to our output range $[0, 1]$ done by taking the absolute value is the best we found. This mapping tends to push both predictions—planar (close to 1) and structural (close to 0)—to each extreme, making them easily separable. Since it is very rare to have the ground truth value in $[-1, 0]$ (corresponds to an obtuse angle), this mapping does not change the pseudo-GT much. With these predictions, the same classification by filtering step with a threshold can be applied to finally label the line as more structural or textural.

3.3. End-to-End Detection and Classification

By applying the data processing method to line segments detected by LSD (Von Gioi et al., 2008), we obtain a dataset with raw images, line segments, and classification labels (using a threshold of 45 degrees) for each segment. This assortment of data, which does not exist yet in any dataset, allows us to train a more complex, end-to-end deep learning model which solves detection and structural-textural classification simultaneously.

To this end, we adopt LETR (Xu et al., 2021)'s architecture. LETR is a state-of-the-art model on wireframe parsing, trained by human-labeled data in Wireframe (Huang et al., 2018) dataset. It works well in capturing wireframe lines depicting large-scale geometry and object shapes. The model's multi-scale encoding-decoding strategy and self-attention mechanism allow it to perform fine-grained line segment detection, and to separate the structural wireframe lines from the background in a binary classification manner. Our adapted model is trained using the same training procedure and loss functions as LETR, but added an additional classification class: now the model classifies the detected lines as structural, textural, or background, instead of deciding between wireframe or background. The best model uses ResNet50 as the backbone and is trained from scratch.



(a) (b)

Figure 3. Performance of different methods on classification by normal differences. (a) The Accuracy-Threshold curve; (b) The ROC curve.

4. Results and Discussions

In this section, we present and discuss the results of our proposed methods. We also demonstrate the usefulness of having this classification result on downstream line-based applications using examples on 3D line reconstruction.

4.1. Classification by Normal Differences

Fig. 2 demonstrates our method's effectiveness. (b)(c) show a comparison of the predicted surface normal differences to the generated pseudo-GT, this value is able to train the model to recover the structural layout. (e) shows the classification results using the models' predictions, which closely recover the labels filtered using the pseudo-GT (in (d)) with the same threshold of 40 degrees.

Model	15°	30°	45°	60°
U-Net, abs	**0.802/0.842**	0.752/0.732	**0.736/0.536**	0.826/**0.254**
FPN, abs	0.797/0.834	**0.757**/0.735	0.726/0.414	**0.833**/0.097
FPN, scale	0.783/0.833	0.753/**0.745**	0.731/0.520	0.828/0.163
Omnidata	0.481/0.228	0.590/0.071	0.701/0.010	0.831/N.A.

Table 1. Accuracy/F1-score of classification by normal differences for each threshold (degree)

The performance of different models and the baseline are compared in Fig. 3 and Tab. 1. The threshold values are used to filter both the GT and predictions. "abs" corresponds to the mapping mentioned in Sec. 3.2, and "scale" is another mapping that linearly maps pseudo-GT range $[-1, 1]$ to $[0, 1]$ and map back at inference. The models are trained for 15 epochs avoiding overfitting. The result shows the models with "abs" mapping leads to the best result, and U-Net and FPN as decoders perform mostly comparably.

We also compare our method with a baseline using the state-of-the-art surface normal estimation model Omnidata (Eftekhar et al., 2021; Kar et al., 2022). The raw

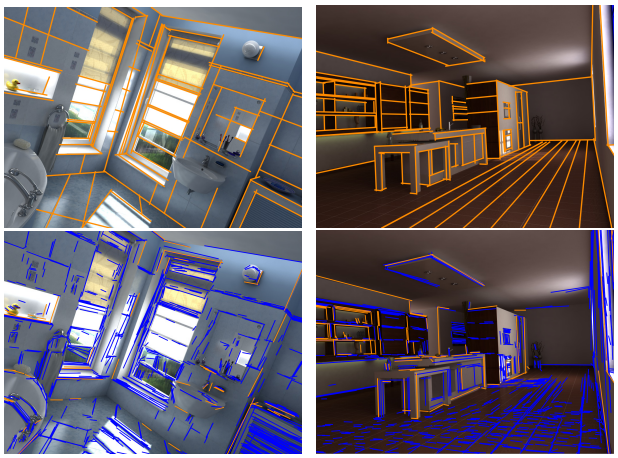


Figure 4. Visualization of detection and classification of LETR. Top: LETR's wireframe detection; Bottom: Our adapted model's detection and structural-textural classification.

images are resized to 384x384 to feed into the model, the output surface normals are rescaled back to the original resolution using nearest neighbor interpolation. The normal differences are then used to obtain the classification in the same way. This baseline performs poorly mainly because it significantly underestimates the normal differences (and leads to the N.A. when the threshold is large), which will be further analyzed in the Appendix. Without the need to explicitly estimate surface normals, our model performs consistently better.

4.2. End-to-End Detection and Classification

Transformer-based models such as LETR require a large amount of data and resources to be trained effectively. Despite limited data and training time, We show our preliminary results adapting LETR to our task in Fig. 4, after training for 425 epochs (200 each for stages 1 and 2, 25 for stage 3). LETR detects wireframe lines consisting of structural and textural lines, while our adaptation demonstrates promising results in both detecting and classifying the two types of lines. This suggests our adaptation could do well on this task, given abundant time and quality data.

4.3. 3D Line Reconstruction with Structural Lines

Having the structural-textural classification on lines could potentially benefit many 3D vision applications such as reconstruction, scene understanding, plane detection, etc. In this project, we specifically explore the effect of using the classification on a yet-to-be-published 3D line reconstruction pipeline. The pipeline takes the detected line segments, does line matching using SOLD2 (Pautrat et al., 2021), and associates 2D line segments into line tracks which each corresponds to one 3D line.

We experimented 2 ways: Filter-before (use only segments labeled as structural to do the reconstruction) and Filter-after (filter the 3D lines reconstructed with all segments by at least 10% of their 2D supports labeled structural). Fig. 5 show the reconstructed lines visible from at least 4 views. Fig. 5(b)(c) show that both ways are able to discard the textural lines (grid patterns on the walls and cupboard, etc.) compared to the baseline reconstruction (d) and present more clearly the structural layout of the 3D scene, while Filter-before (b) is stricter and keeps only more structural lines than Filter-after (c). The shown visualization used a filtering threshold of 30 degrees. Changing the degree values could lead to stricter or looser filtering of textural lines depending on the use case and scenes. We believe the cleaner reconstructions of 3D scenes could benefit tasks like scene understanding and semantic segmentation by removing interference from the many textural lines, the kept structural lines could also be used for interesting tasks like plane detection.

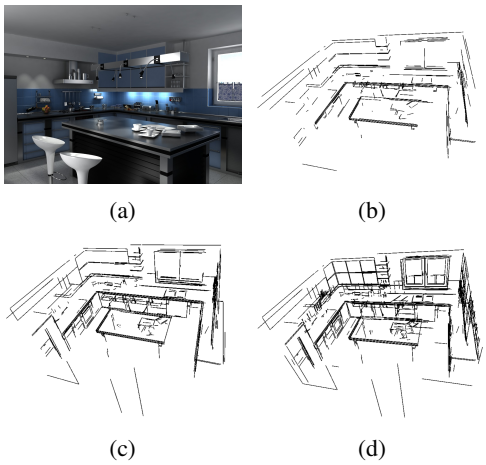


Figure 5. Visualization of 3D line reconstructions. (a) Image of the 3D scene; (b) Filter-before (use only segments labeled as structural); (c) Filter-after (filter reconstructed 3D lines by 2D supports); (d) Reconstruction with all segments. Filtering threshold: 30 deg.

5. Conclusion

In this project, we present the first-ever attempt at the novel task of classifying line segments detected on images by whether they represent more of a structural line or a textural line in the 3D scene. We note that due to the small amount of data and training time, our models' performances have large space for improvements, but are promising in both our proposed per-pixel estimation and filtering, and end-to-end detection and prediction methods. We also demonstrate the usefulness of such classification on typical vision task. Future directions include integrating depth information to solve cases surface normal cannot distinguish, gathering more labeled data, and further exploring end-to-end solutions. We believe there are many to explore for this task and it will be very beneficial for computer vision.

References

Akinlar, C. and Topal, C. Edlines: Real-time line segment detection by edge drawing (ed). In *IEEE International Conference on Image Processing*, 2011.

Bresenham, J. E. Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1):25–30, 1965.

Chen, W., Xiang, D., and Deng, J. Surface normals in the wild. In *International Conference on Computer Vision (ICCV)*, 2017.

Eftekhar, A., Sax, A., Malik, J., and Zamir, A. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *International Conference on Computer Vision*, 2021.

Gao, S., Wan, J., Ping, Y., Zhang, X., Dong, S., Li, J., and Guo, Y. Pose refinement with joint optimization of visual points and lines. *arXiv preprint arXiv:2110.03940*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hofer, M., Maurer, M., and Bischof, H. Efficient 3D scene abstraction using line segments. *Computer Vision and Image Understanding (CVIU)*, 157:167–178, 2017.

Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., and Ma, Y. Learning to parse wireframes in images of man-made environments. In *Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Huang, S., Qin, F., Xiong, P., Ding, N., He, Y., and Liu, X. Tp-lsd: Tri-points based line segment detector. In *European Conference on Computer Vision (ECCV)*, 2020.

Kar, O. F., Yeo, T., Atanov, A., and Zamir, A. 3d common corruptions and data augmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. Feature pyramid networks for object detection. In *Computer Vision and Pattern Recognition (CVPR*, 2017.

Pautrat, R., Lin, J.-T., Larsson, V., Oswald, M. R., and Pollefeys, M. Sold2: Self-supervised occlusion-aware line description and detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., and Moreno-Noguer, F. PL-SLAM: Real-time monocular visual SLAM with points and lines. In *International Conference on Robotics and Automation (ICRA)*, 2017.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 2020.

Ranftl, R., Bochkovskiy, A., and Koltun, V. Vision transformers for dense prediction. In *International Conference on Computer Vision (ICCV)*, 2021.

Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M. A., Paczan, N., Webb, R., and Susskind, J. M. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *International Conference on Computer Vision (ICCV)*, 2021.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Von Gioi, R. G., Jakubowicz, J., Morel, J.-M., and Randall, G. Lsd: A fast line segment detector with a false detection control. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 32(4):722–732, 2008.

Xu, C., Zhang, L., Cheng, L., and Koch, R. Pose estimation from line correspondences: A complete analysis and a series of solutions. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 39(6), 2017.

Xu, Y., Xu, W., Cheung, D., and Tu, Z. Line segment detection using transformers without edges. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

Xue, N., Wu, T., Bai, S., Wang, F., Xia, G.-S., Zhang, L., and Torr, P. H. Holistically-attracted wireframe parsing. In *Computer Vision and Pattern Recognition (CVPR)*, 2020.

Yoon, S. and Kim, A. Line as a visual sentence: Context-aware line descriptor for visual localization. *IEEE Robotics and Automation Letters (RAL)*, 6(4):8726–8733, 2021.

Zhang, Z., Li, Z., Bi, N., Zheng, J., Wang, J., Huang, K., Luo, W., Xu, Y., and Gao, S. PPGNet: Learning point-pair graph for line segment detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

Zhou, Y., Qi, H., Zhai, Y., Sun, Q., Chen, Z., Wei, L.-Y., and Ma, Y. Learning to reconstruct 3D manhattan wireframes from a single image. In *International Conference on Computer Vision (ICCV)*, 2019.

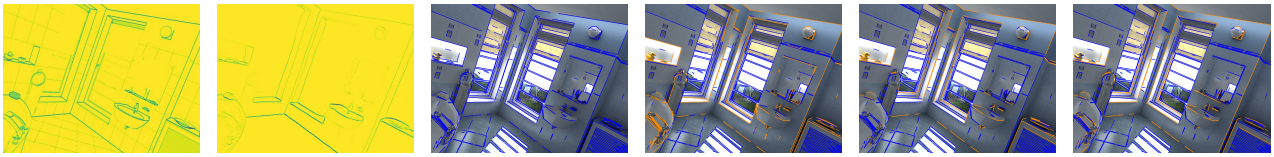
Zuo, X., Xie, X., Liu, Y., and Huang, G. Robust visual slam with point and line features. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017.

A. Appendix

A.1. Normal Difference Regression with Weighted Loss and Focal Loss

In our experiments on per-pixel normal difference predictions, we observed that the normal difference estimations tend to overestimate when the normal changes largely and underestimate other places. We then sought to weighted loss and focal loss to try to improve. The weighted loss is done by weighting up the loss contribution from pixels in the vicinity of detected lines, and focal loss is applied to let the model focus more on pixels with large loss. Unfortunately, the results are not very conclusive, but the functions are kept in our submitted code.

A.2. Analysis of Omnidata Baseline Results

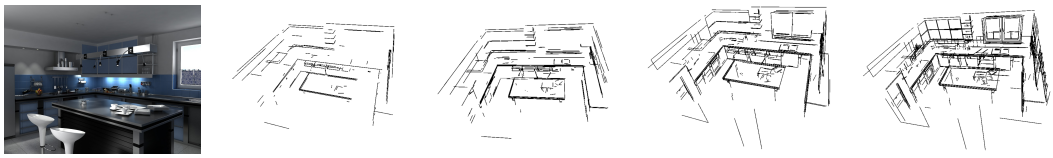


(a) (b) (c) (d) (e) (f)

Figure 6. (a) Generated pseudo-GT normal differences; (b) Normal differences computed from Omnidata output; (c) Classification on (b) with threshold 40 degrees; (d) Classification on (b) with threshold 0.999 in cosine value; (e) Classification using our model with threshold 40 degrees; (f) Classification using pseudo-GT with threshold 40 degrees.

As mentioned in Sec. 4.1, the Omnidata (Eftekhar et al., 2021; Kar et al., 2022) model significantly underestimates the angle between surface normals and therefore the per-pixel surface normal differences, as shown in Fig. 6(b) comparing to Fig. 6(a). This leads to the complete failing of filtering in (c) with the same threshold as (e) and (f). With an extreme threshold of 0.9999 (cosine value) as shown in (d), the classification finally gets close to that of pseudo-GT and our model, but still shows many errors. This extreme threshold is also case-dependent: as the estimated surface normals are of high variance, this threshold will not work in some other cases. This again shows our model's novelty and effectiveness: surpassing the step of explicitly estimating the surface normals lead to stably better performance.

A.3. Effect of Filtering Threshold in 3D Line Reconstruction



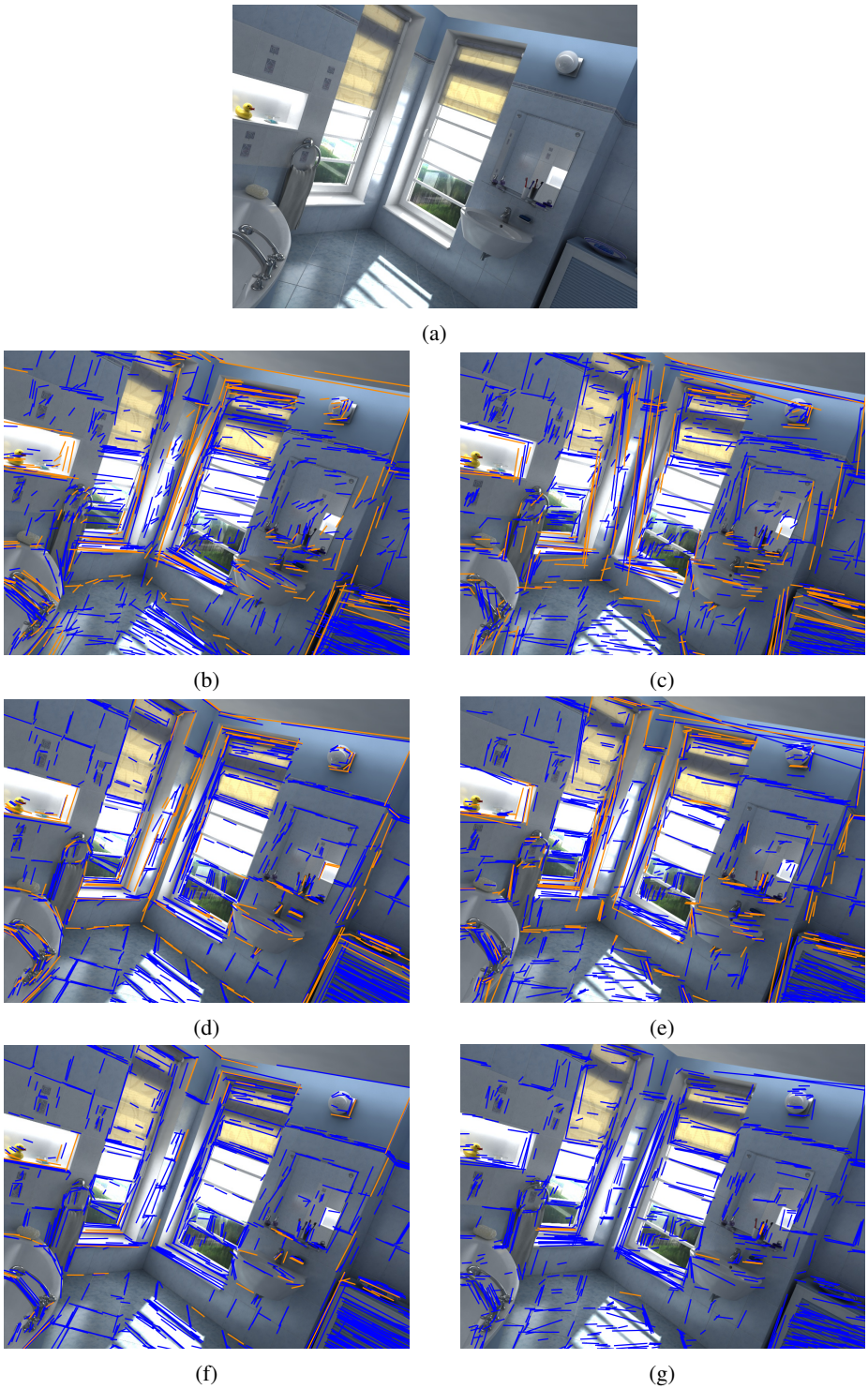
(a) (b) (c) (d) (e)

Figure 7. 3D Line Reconstruction by Filter-after with different thresholds. (a) Image of the 3D scene; (b) Threshold: 40 degrees; (c) Threshold: 35 degrees; (d) Threshold: 30 degrees; (e) No filtering, using all line segments.

Here we present the visualization of using the 3D line reconstruction pipeline mentioned in Sec. 4.3 with different threshold values. The shown results are all produced with Filter-after strategy, as Filter-before consistently enforces stricter filtering on the lines than Filter-after with the same threshold. As shown in Fig. 7, the threshold value greatly affects the number of resulting 3D lines. With a tighter threshold (larger angle in degrees), the more structural lines are kept and less structural lines are filtered out. We argue that this adds to the benefits of having structural-textural classification/filtering for downstream tasks. As shown here, by adjusting the threshold, different levels of results could be used for the tasks depending on the use case and scene.

A.4. Stages Comparison in Adapted LETR Training

We show here the visualization of the output of the three stages during the training of our adapted LETR model in Fig. 8.



(a) (b) (c) (d) (e) (f) (g)

Figure 8. Stages Comparison in Adapted LETR Training. (a) Input image; (b) Output of stage 1 with backbone ResNet50; (c) Output of stage 1 with backbone ResNet101; (d) Output of stage 2 with backbone ResNet50; (e) Output of stage 2 with backbone ResNet101; (f) Output of stage 3 with backbone ResNet50; (g) Output of stage 3 with backbone ResNet101.