Spatially Multi-conditional Image Generation
Supplementary material

Nikola Popovic[1*] Ritika Chakraborty[1*] Danda Pani Paudel[1] Thomas Probst[1]
Luc Van Gool[1,2]
[1]Computer Vision Laboratory, ETH Zurich, Switzerland
[2]VISICS, ESAT/PSI, KU Leuven, Belgium
{nipopovic,critika, paudel, probstt, vangool}@vision.ee.ethz.ch

This document provides additional details, which complement the main paper. We provide the complete network diagram of our generator in Section 1. More implementation details are contained in in Section 2. The visualization of input labels under different sparsity can be found in Section 3. In Section 4, we demonstrate an intuitive application of our method. Additional qualitative results are presented in Section 5. Finally, a discussion about ethical and societal impacts is contained in Section 6. Also, please refer to our video supplementary for the example of geometric editing.

1. Generator Overview

Whole Network. The network diagram of our generator is presented in Figure 1. The figure shows how our proposed Label Merging Block TLAM is connected to the ASAP-Net [4] generator, which we use as the backbone.

ASAP-Net Generator. The generator takes the *Concept Tensor* $\mathsf{Z} \in \mathbb{R}^{H \times W \times d}$ as an input, similar to taking the semantic labels in the original design. It then outputs a tensor of weights, which parameterize the pixelwise spatially-varying multi-layer perceptrons (MLPs). The MLPs, with infered weights, compute the final output image by taking the *Concept Tensor* Z as their input.

2. Implementation Details

Projection Block. The Projection Block projects each input label into an embedding space with the same dimensionality. Every input label is processed by one 1x1 convolution layer, followed by a nonlinear activation. The projection is a 96-dimensional tensor for each label.

Concept generation Block. The Concept Generation Block takes the projected tensors as input, and operates over them in a pixel-wise fashion. For each pixel, we thus have an embedding vector for each task. We add a label specific-encoding to the embedding vector of each label, as a way of signalizing which task the embedding corresponds to. For every pixel, the concept generation block processes each set of encoded labels with a transformer. The transformer consists of 3 layers, where each uses 3 heads.

[*]Equal contribution.

3. Visualization of input Labels

In Figure 2 we show examples of the input labels from the Taskonomy dataset. These labels include semantic segmentation, normals, depth, edges and curvature. Furthermore, in Figure 3, 4 and 5 we show examples of labels from the Taskonomy dataset, with 70%, 50% and 30% label sparsity. To generate sparse labels, we look at spatial areas corresponding to distinct semantic segmentation instances. For the sparsity of S, we randomly drop the labels with $S\%$ probability, independently for every label inside every area. Higher sparsity means there is a higher probability that a semantic region will be masked out for all labels.

4. Geometric Image Editing with User Inputs

In order to demonstrate an intuitive application of our method, we perform image editing by inserting a new object into the scene. Figure 8 shows how our method can mimic rendering while allowing the geometric manipulation of an image. In this application, we render a table in the given image, by simply augmenting different labels to include label information derived from a 3D model. Our method is able to render the table realistically within the image, while ASAP and SPADE perform unsatisfactorily. Figure 9 shows an example of removing a certain object from the image by augmenting the labels.

5. Additional Qualitative Results

In Figure 6 we show images synthesized with our TLAM model using dense input labels (semantics, curvature, edges, normals and depth), on the Taskonomy dataset. In Figure 7 we show images synthesized with our

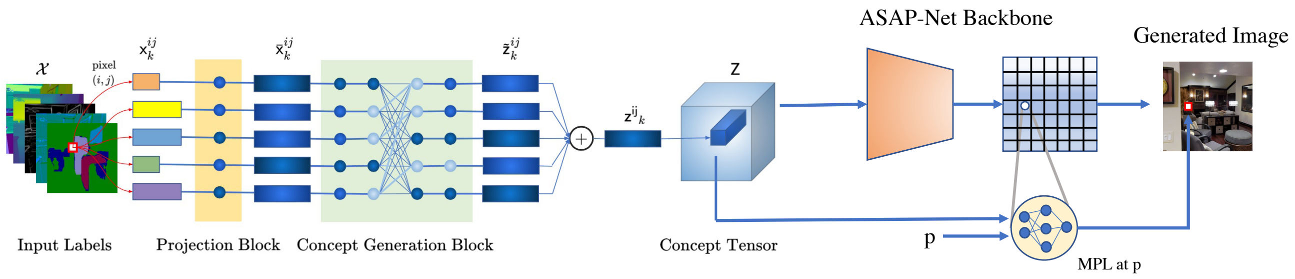

Figure 1. **Label Merging TLAM block and ASAP-Net Backbone.** The input labels are processed by the label merging network to output the *Concept Tensor*. The ASAP generator takes the *Concept Tensor* $\mathsf{Z} \in \mathbb{R}^{H \times W \times d}$ as an input and outputs a tensor of weights. Those weights are parameters of pixelwise, spatially-varying, MLPs, which compute the final output image from the *Concept Tensor* Z.

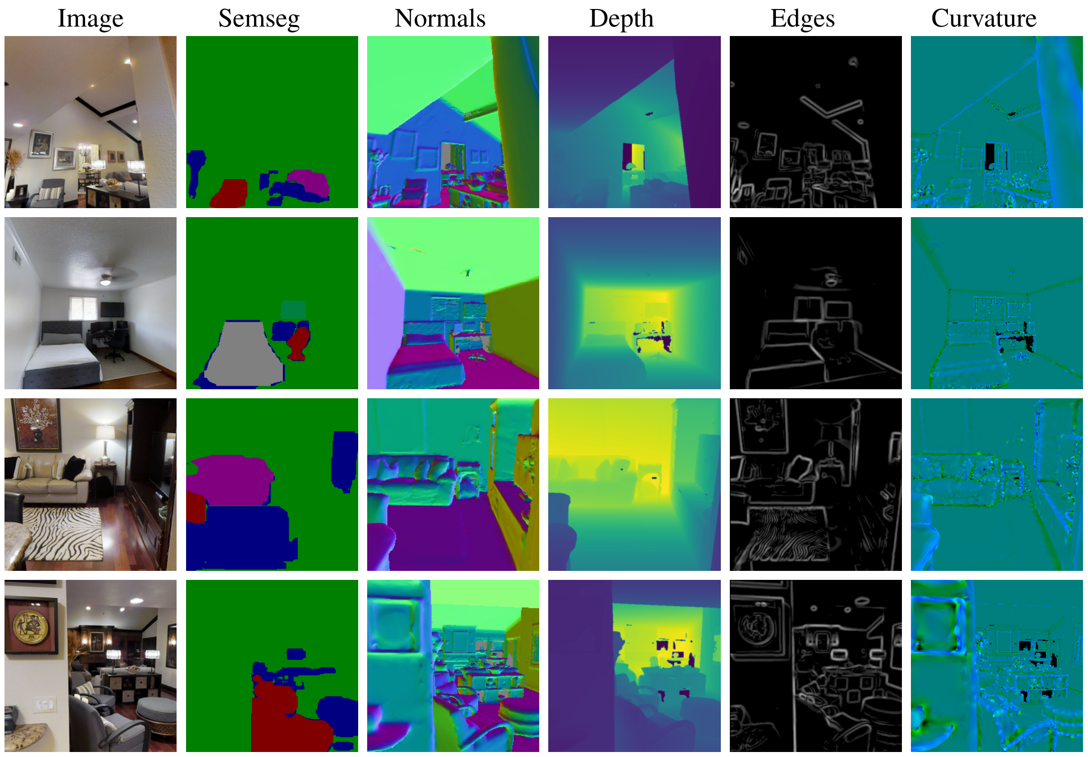

Figure 2. **Examples of dense input labels.** Samples of the Taskonomy dataset with all input labels visualized.

TLAM model using sparse input labels (50% sparsity), on the Taskonomy dataset. Finally, Figure 10 shows additional visual comparison on the Cityscapes dataset, where we compare our TLAM and Sparse-TLAM methods with SPADE [3] and ASAP-Net [4].

6. Ethical and Societal Impact

This work is going one step further into the image generation. While bringing great potential artistic value to the general public, such technology can be misused for fraudulent purposes. Despite the generated images looking realistic, this issue can be partially mitigated by learning-based fake detection [2, 6, 5].

In regards to limitations, our method is not designed with a particular focus on balanced representation of appearance and labels. Therefore, image generation may behave unexpected on certain conditions, groups of objects or people. We recommend application-specific strategies for data collection and training to ensure the desired outcome [1, 7].

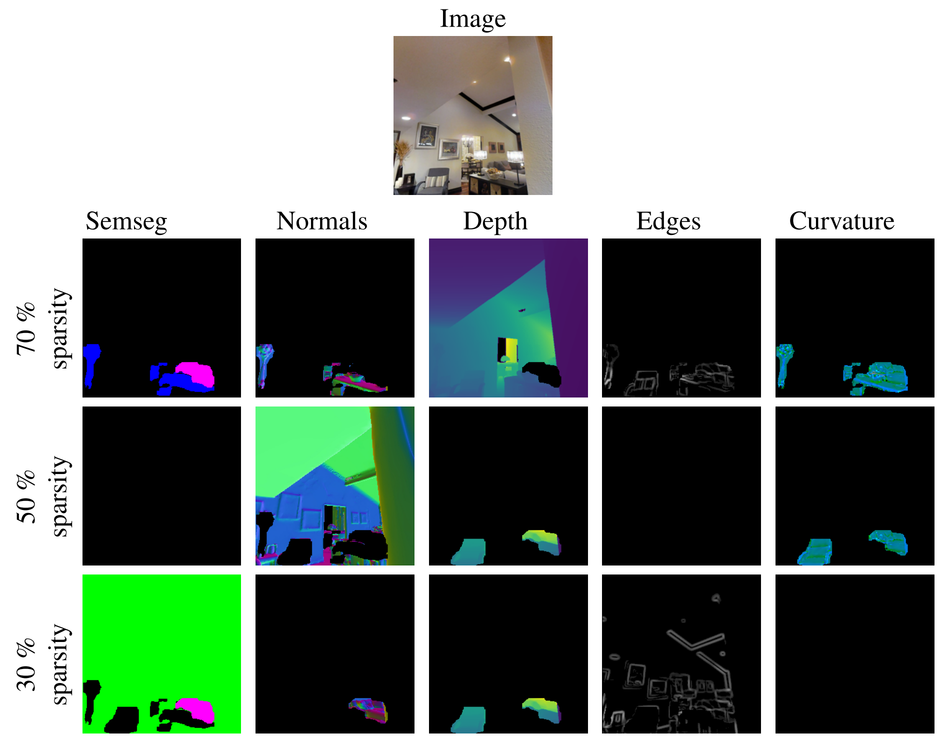

Figure 3. **Example of sparse labels.** One sample from the Taskonomy dataset where input labels have the sparsity of 70%, 50% and 30%.

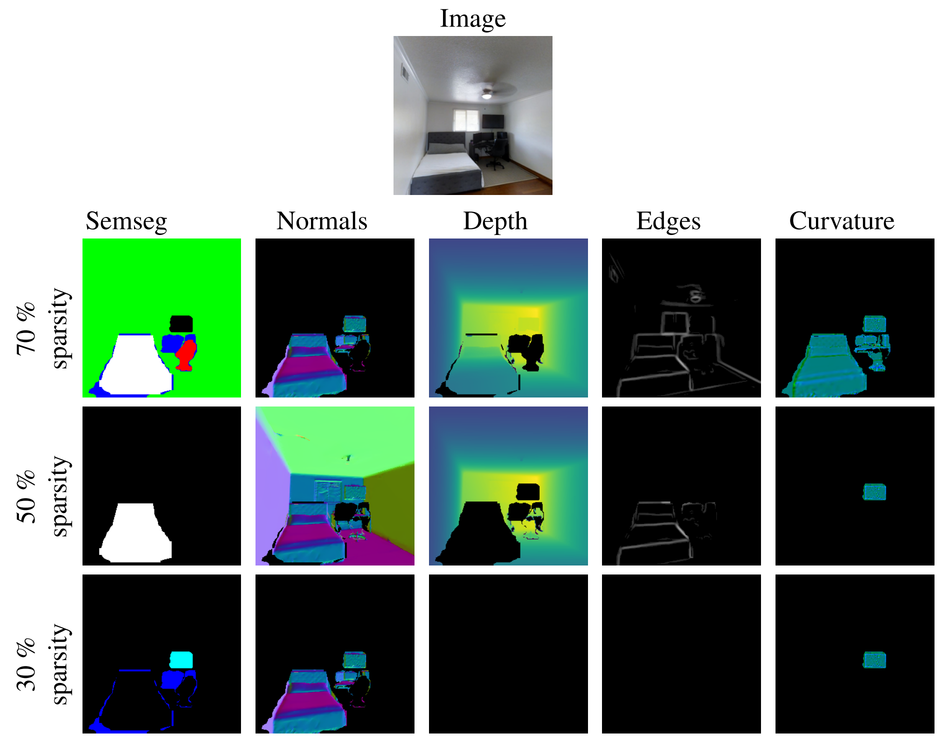

Figure 4. **Example of sparse labels.** One sample from the Taskonomy dataset where input labels have the sparsity of 70%, 50% and 30%.

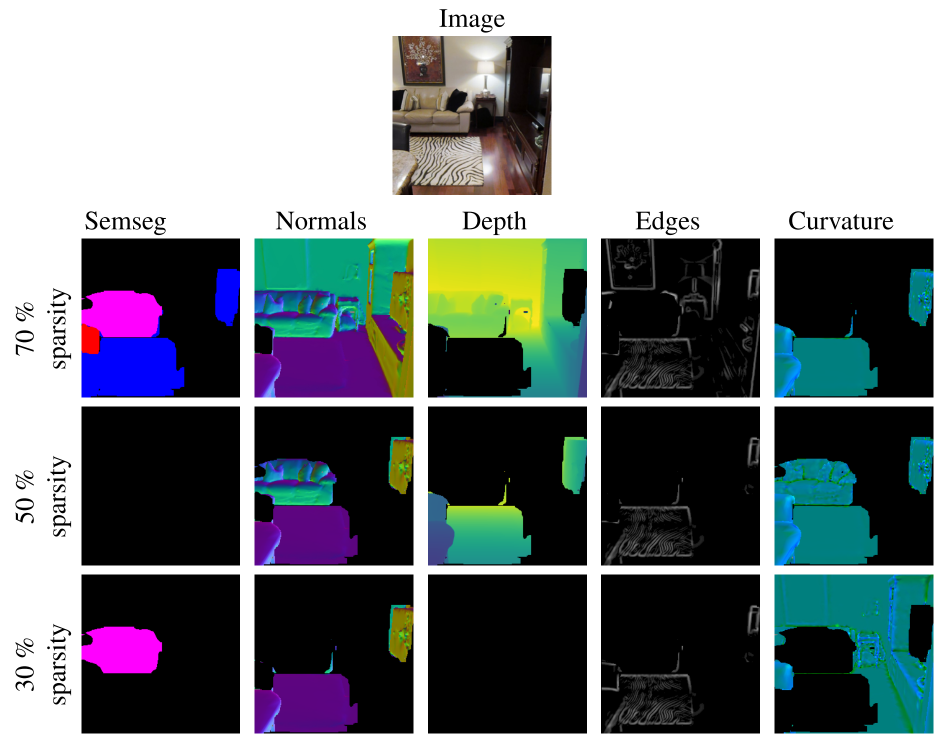

Figure 5. **Example of sparse labels.** One sample from the Taskonomy dataset where input labels have the sparsity of 70%, 50% and 30%.

References

[1] Mohsan S. Alvi, Andrew Zisserman, and Christoffer Nellåker. Turning a blind eye: Explicit removal of biases and variation from deep neural network embeddings. In *ECCV Workshops*, 2018.

[2] Luca Guarnera, Oliver Giudice, and Sebastiano Battiato. Deepfake detection by analyzing convolutional traces. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2841–2850, 2020.

[3] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization, 2019.

[4] Tamar Rott Shaham, Michael Gharbi, Richard Zhang, Eli Shechtman, and Tomer Michaeli. Spatially-adaptive pixelwise networks for fast image translation. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[5] Luisa Verdoliva. Media forensics and deepfakes: An overview. *IEEE Journal of Selected Topics in Signal Processing*, 14:910–932, 2020.

[6] Shengyu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A. Efros. Cnn-generated images are surprisingly easy to spot… for now. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8692–8701, 2020.

[7] Zeyu Wang, Klint Qinami, Yannis Karakozis, Kyle Genova, Prem Qu Nair, Kenji Hata, and Olga Russakovsky. Towards fairness in visual recognition: Effective strategies for bias mitigation. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8916–8925, 2020.

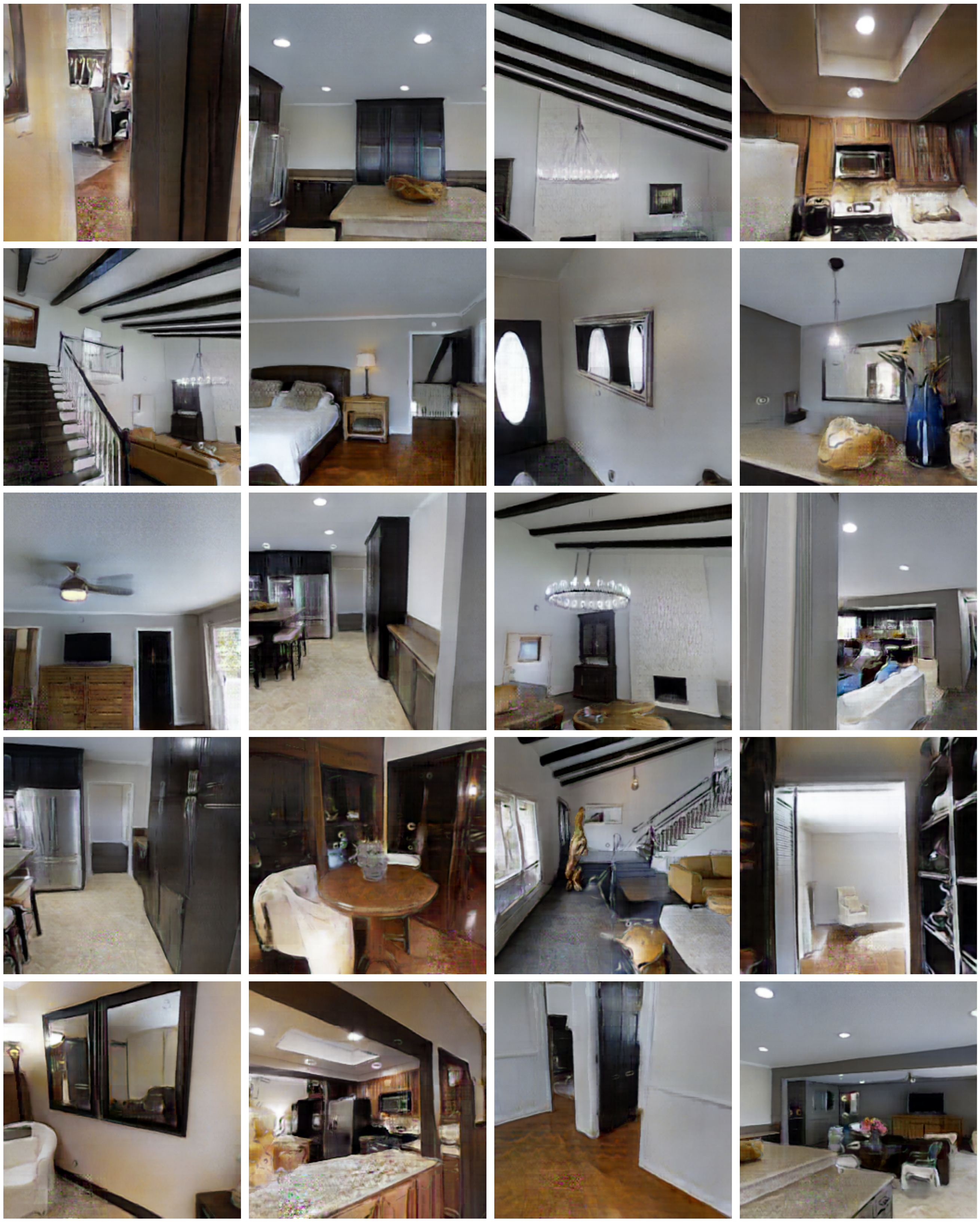

Figure 6. **Images generated using dense input labels.** Here we see images generated with our proposed TLAM label merging model, using dense input labels from the Taskonomy dataset.

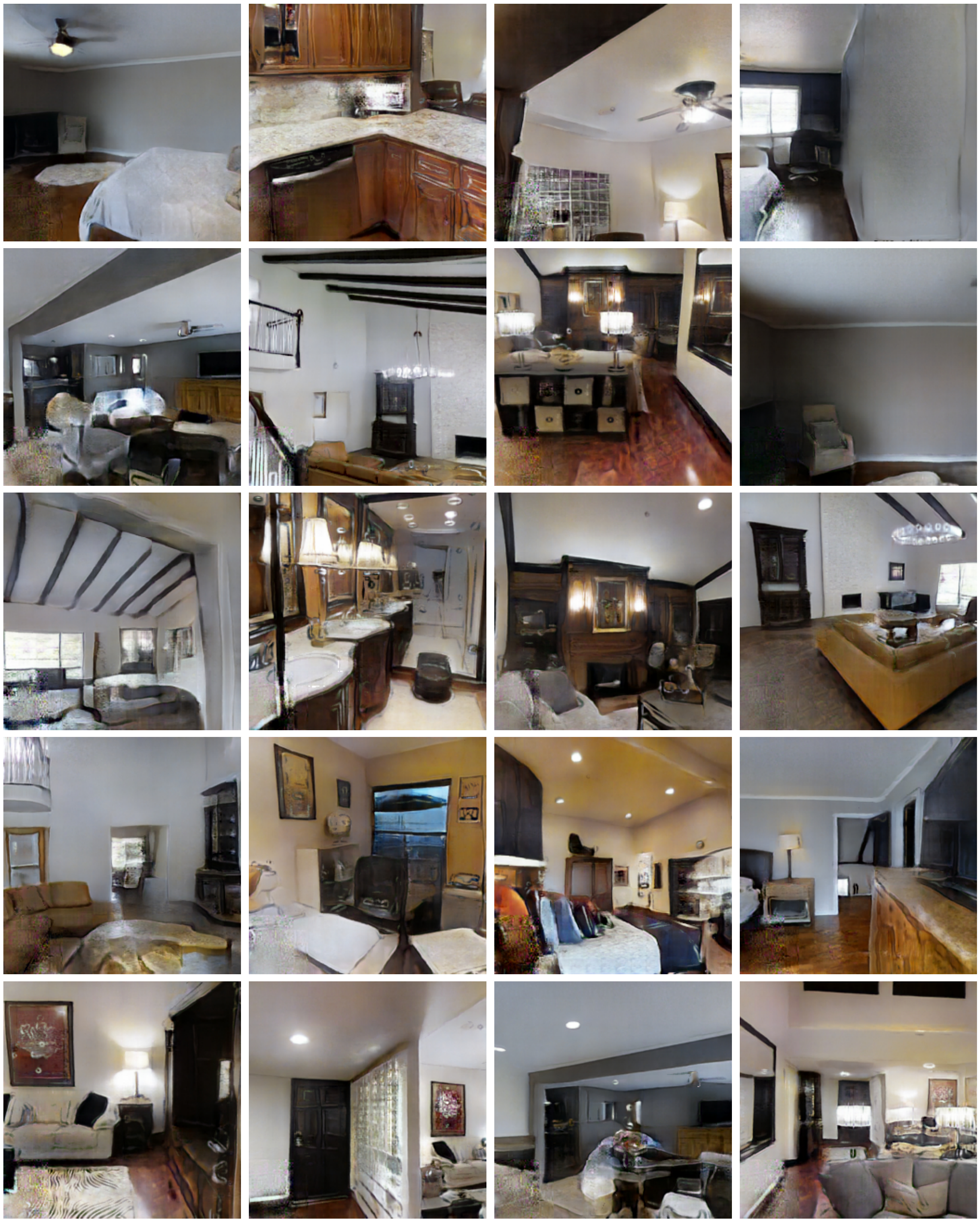

Figure 7. **Images generated using 50% sparse input labels.** Here we see images generated with our proposed TLAM label merging model, using input labels from the Taskonomy dataset with 50% sparsity.

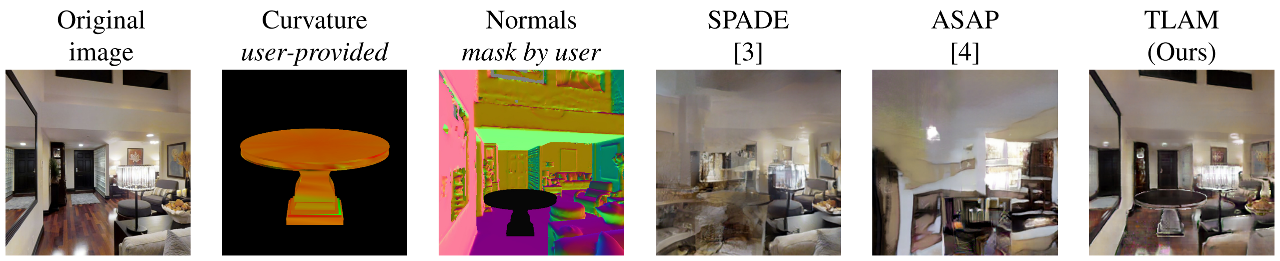

Figure 8. **Geometric image editing with user inputs.** From left to right: original image; curvature label (provided among five) of the table to be inserted into the image; semantics of the table on the normal map of the scene; generated image using SPADE, ASAP and our method, respectively. During editing, we use dense labels of the original image, where we introduce five labels derived from a texture-less 3D model of a table (object of interest)– labels are first edited to introduce the object, followed by the image generation using the proposed TLAM method.

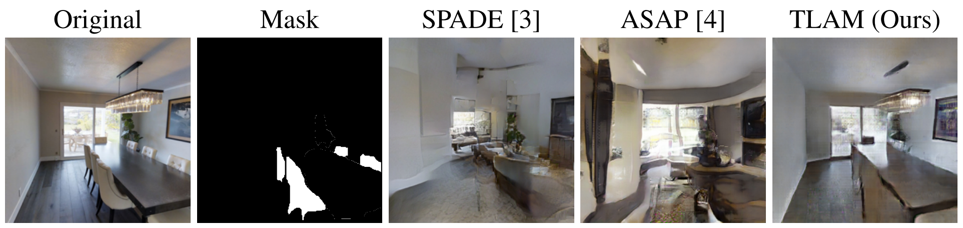

Figure 9. **Object removal.** From left to right: original image; mask of the removed object; generated image using SPADE, ASAP and our method, respectively. This figure shows removal of chairs from the image, by removing their labels. The mask for removal is *manually* chosen by the user.

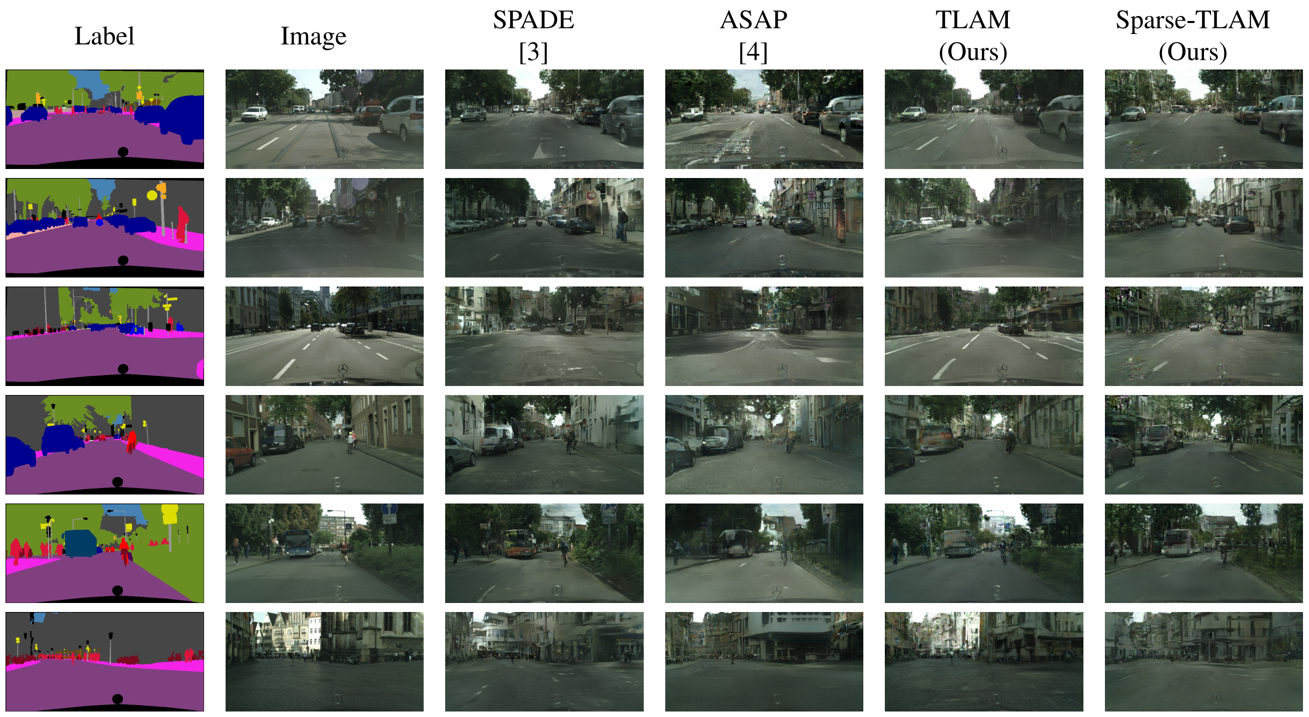

Figure 10. **Image generation of different methods on Cityscapes.** Here we compare generated results of our methods TLAM and Sparse-TLAM to SPADE and ASAP-Net.