Plan2Scene: Converting Floorplans to 3D Scenes
Supplemental Materials

Madhawa Vidanapathirana Qirui Wu Yasutaka Furukawa Angel X. Chang Manolis Savva
Simon Fraser University

A. Additional Implementation Details

A.1. Vectorization and 3D Geometry Construction

Floorplans are most commonly available online in raster image formats. To convert such images to vector floorplans we can employ prior work such as the Raster-to-Vector system of Liu et al. [6]. The vector floorplan outputs delineates walls, doors and windows as line segments defined by 2D coordinate pairs, room type labels of each room and positions of fixed objects (e.g., sinks, toilets, kitchen islands). In this paper, we do not focus on the floorplan vectorization stage of the Plan2Scene task, so we use ground truth vectorized floorplans provided in our dataset, as described in the main paper.

To convert the vectorized floorplan to 3D geometry we employ a simple rule-based approach. The Rent3D dataset separately defines the wall boundary of each room as a polygon consisting of wall line segments, with doors and windows of each room also specified by line segments. We assign each door/window annotation line segment to the closest wall line using a threshold distance of 20 pixels and a threshold direction vector angle difference of $\arccos(0.9)$ (i.e. dot product of direction vectors greater than 0.9).

We then generate 3D geometry for each room by extruding the architectural surfaces (walls, floor and the ceiling) with a number of default dimension values. The scale factor to real-world measurement units is available in the floorplan dataset we use. Wall height is conditioned on the room type, with all walls set to 2.8m tall, except for walls incident on balcony exterior at 1.4m. Wall thickness is set to 0.1m. Then, holes are created in the walls for doors and windows. The height and vertical placement of holes are determined by the 3D mesh model chosen for door or window (see next section).

A.2. 3D Object Placement

We choose suitable 3D object models for doors, windows and other fixed objects from the ShapeNet [2] dataset using a rule-based algorithm.

Placing 3D models for doors and windows. For each door and window, we first select appropriate 3D mesh models.

(a) Entrance door with paving at bottom.

(b) Bathroom window placed at higher elevation.

(c) Closets next to each other, precisely fitting the AABBs.

(d) Kitchen objects 'clamped' to closest wall.

Figure 1: Illustration of object placement handling in a variety of scenarios, as specified by rules in Table 1.

We used 26 representative door and window models from ShapeNet [2]. The 3D mesh models are aligned, metrically scaled, and tagged with appropriate metadata to allow selection conditional on: 1) opening type (door or window), 2) exterior vs interior use (e.g., entrance door with floor paving shown in Figure 1a is used for exterior doors) and 3) opening for particular room types (e.g., balcony doors, or high windows for bathrooms as in Figure 1b). Among the object models that meet the selection criteria, we choose the one closest in length to the width of the opening in the floorplan. We scale and orient the CAD model so as to fit the opening width.

Table 1: Rules used to place 3D mesh models for fixed object icons indicated on the input floorplans.

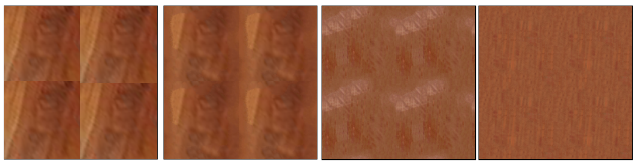
		Closest wall		Object Icon AABB	
Category	Selection criteria	Clamp	Shrink to fit	Orient	Fit width
`closet`	AABB size			✓	✓
`sink`	room type	✓			
`bathtub`	wall size	✓	✓		
`toilet`	-	✓			
`cooker`	-	✓	✓		

Placing 3D models for fixed objects. Icons indicating placements of fixed objects are prevalent in floorplans. We manually annotated the Axis Aligned Bounding Boxes (AABBs) of 'cooker', 'sink', 'bathtub', 'toilet' and 'closet' indicated on our test set floorplans. Then, we used 22 3D mesh object models from ShapeNet [2] to populate these icons. All these objects are scaled to real world units and consistently aligned. The first step in placement is selection of a specific 3D mesh variation. Table 1 shows a set of criteria based on room category, width of annotation AABB, and size of closest wall, which we used to determine the most suitable 3D mesh model to place. The closet in Figure 1c illustrates selection of an object model that is closest to the width of the icon AABB.

Table 1 also summarizes the rules used for determining the positioning, scaling and orientation of fixed objects. By default, we position each object centered at the indicated placement on the input floorplan. If the 'clamp to closest wall' rule is set, the object is moved and rotated to ensure the backside of the object abuts the wall. Figure 1d shows an example of a cooker and a kitchen sink clamped to a wall. If the 'orient to object icon AABB' rule is set, the object is rotated to fit the larger dimension (width) of the AABB, additionally using two ray-tests to face the backside against the nearby wall. We send two rays orthogonal to the larger dimension, originating from the center of AABB. In some cases, we need to re-scale the object based on the size of the icon or the surrounding space (e.g., matching the width of the closet icon, or shrinking to provide clearance when clamping against a wall).

A.3. Photo Room Assignment

We did not focus on the photo-to-room assignment stage of the Plan2Scene task. Assignments of photos to rooms are sometimes found in the online real estate listings. These assignments can also be carried out as a simple annotation task, or be automated through scene classification-based methods. In this paper, we use ground truth photo assignments as provided by our dataset. This stage is certainly an interesting research problem that can be investigated in future work. Prior work has also focused specifically on aspects related to the photo assignment problem and may be utilized and extended to address this problem [5].



(a) Surface crop (b) Seam corrected crop (c) Non-stationary texture (d) Stationary texture

Figure 2: Illustration of requirements for texture to be tileable. Sub-figure (a) is a rectified surface crop, which is 2x2 tiled. Here, the discontinuity between crops is a tiling artifact. Sub-figure (b) is a 2x2 tiling of the same crop, after seam correction. Here, the pixels at the border of the crop are modified to remove seams. Another requirement is for the texture crops to be stationary. Sub-figures (b) ,(c) and (d) are all seam corrected. Yet, in sub-figures (b) and (c), repetitiveness induced by tiling is visible due to non-stationary. Sub-figure (d) satisfies both requirements and does not produce undesirable tiling artifacts.

A.4. Texture Synthesis Module

Tileable textures. Figure 2 shows visual examples of the steps necessary to produce a tileable texture from an image patch. A rectified surface crop will have discontinuities when tiled. By applying a seam correction post-process, the discontinuity at the seams is smoothed out. We also need to ensure that the generated texture is stationary to avoid repeating patterns due to non-stationarity in the texture.

Median color computation. When computing the median color of a surface crop, we separately compute the median of pixel values for each RGB color channel. The median color is then the color represented by these three RGB medians.

A.5. Texture Propagation (GNN) Module

Graph representation. The Rent3D dataset separately defines doors for each room. Thus, inferring room-door-room connectivity requires pairing of door line segments from adjacent rooms. We pair door line segments that belong to two rooms using a threshold distance of 20 pixels and a threshold direction vector angle difference of $\arccos(0.9)$ (dot product of direction vectors greater than 0.9). Then, a pair of rooms that shares a door pair is connected by an edge in our graph representation.

GNN architecture details. Our GNN Architecture consists of the following layers: Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Gated Graph Convolution $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Linear. The input dimension is d and the output dimensions is 3×11. The intermediate layers handle $2d$-dimensional vectors. The Gated Graph Convolution Layer has a sequence length of 3.

Training curriculum. Recall that we train the GNN by taking each observed surface in the training set as unobserved, and then treating it as a prediction target. Therefore, for each observed surface s of every room r in every training set house, we create a graph $G_{r,s}$ which is a copy of the room-door-room connectivity graph G of that house, having masked the surface texture embedding $\vec{t}^*_{r,s}$ from the input. The masking involves zeroing the surface texture embedding and the corresponding "presence cell".

Graph augmentation. We replace each graph instance $G_{r,s}$ generated above by 7 mutated graphs $\{G^1_{r,s}, \ldots G^j_{r,s}, \ldots, G^7_{r,s}\}$ where additional surfaces are masked subjected to different probabilities. The criteria used generates 3 copies with no additional mutations, 2 copies where a surface is unobserved subjected to a probability of 0.2, and one copy each where a surface is unobserved subjected to probabilities 0.4 and 0.6. We use these mutated graphs to train the GNN.

Prediction targets for training. The GNN computes 3 surface texture embeddings for each node: the floor texture embedding $\hat{t}_{r,1} \approx \vec{t}^*_{r,1}$, the wall texture embedding $\hat{t}_{r,2} \approx \vec{t}^*_{r,2}$ and the ceiling texture embedding $\hat{t}_{r,3} \approx \vec{t}^*_{r,3}$. The training targets are embeddings $\vec{t}^*_{r,1}$, $\vec{t}^*_{r,2}$ and $\vec{t}^*_{r,3}$ computed by our improved texture synthesis approach for observed surfaces leveraging the textureness score. The GNN is only used for inferring texture embeddings of unobserved surfaces. Therefore, we do not encourage training signals that represent application of GNN for an observed surface. Hence, for each training graph $G^j_{r,s}$ the loss function is applied only to the output $\hat{t}_{r,s}$, which is the texture embedding prediction corresponding to the surface s of room r. Our training curriculum ensures this surface is unobserved.

B. Substance-Mapped Textures Dataset

In the main paper we describe the collection of tileable (seamless and stationary) textures that we use to power the texture retrieval baseline against which we compare our approach. Figure 3 shows several example textures from each substance category of this dataset. This dataset contains a total of 146 high quality textures that are directly tiled onto architectural surfaces by the retrieval baseline.

C. Details of SUBS Metric

This metric measures the substance classification error rate. It applies a substance classifier separately on both the generated texture and the reference crop. If the two predictions do not match, then we count this as a substance error. Details of the substance classifier we use for this metric are as follows. We re-initialized and trained the last linear layer of a VGG16 network pretrained on ImageNet. The training set is a mix of crops sampled from our stationary textures dataset and crops sampled from rectified surfaces in OpenSurfaces [1] (which provides substance categories). We extracted twenty 128x128 crops per texture, from a subset of textures in the train split of the stationary textures dataset and up to ten 128x128 opaque crops per surface from a subset of rectified surface masks provided by OpenSurfaces. This resulted in a train set of 10498 crops (4120 from stationary textures dataset and 6378 from OpenSurfaces). The dataset covered four substances ('wood', 'painted walls', 'carpet' and 'tiles'). We used weighted cross-entropy loss for training, where each class label is weighted by the L2 Norm(1/(1+'frequency of label in the train set')). The trained model has a accuracy of 87.9% on an unseen validation set of 60 crops extracted from OpenSurfaces and 64 crops extracted from our stationary textures dataset.

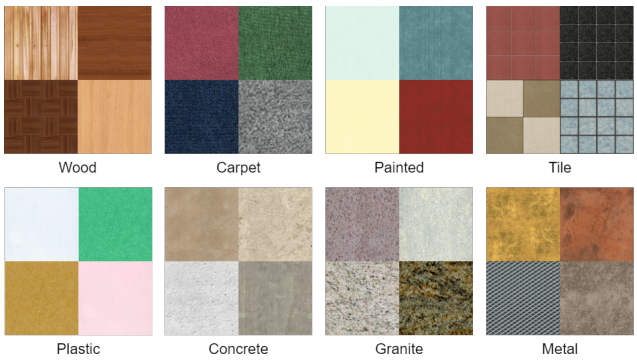


Figure 3: Selected samples from Substance Mapped Textures dataset.

D. Additional Quantitative Evaluation

How does performance depend on unobserved photo fraction? Figure 4 shows the performance of various methods on all surfaces at different levels of simulated unobserved photo fractions. Figure 5 plots the same results on unobserved surfaces. Our `Synth` approach performs the best across the spectrum among all the tileable approaches.

How similar is the generated texture distribution to the input? We use FID [4] to compute the similarity in distribution between the generated textures and the surface crops. For our method `Synth`, the FID scores of observed surfaces (196.1), unobserved surfaces (199.4) and the overall FID score considering all the surfaces (196.2) are roughly the same. This indicates that both predictions for unobserved surfaces by the GNN and predictions for observed surfaces using the textureness score have an equal degree of similarity to the input surface crops. Table 2 computes FID between generated textures and surface crops for different surface types and room types. Similarly to the overall FID scores across all surfaces, `Synth` performs the best among all tileable approaches. We can conclude that we do better

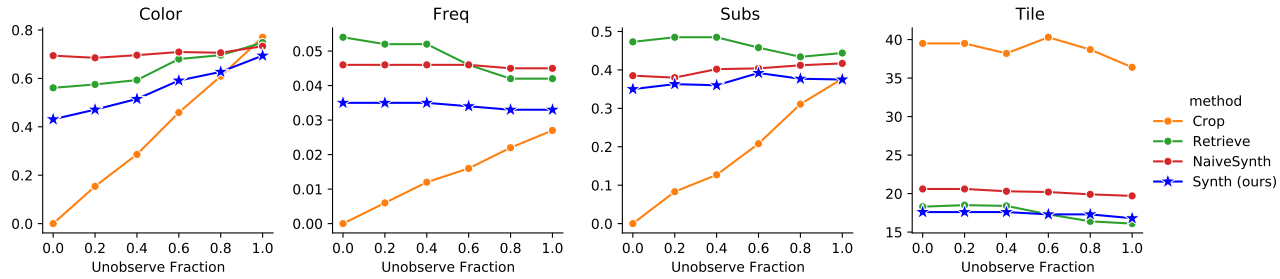


Figure 4: Evaluation on the test set across a spectrum of simulated unobserved photo fractions. Lower metric values are better. Fraction equal to 0 indicates no simulated unobserved photo, whereas 1.0 indicates removal of all photos. `Synth` (ours) is the best performing tileable approach across the spectrum.

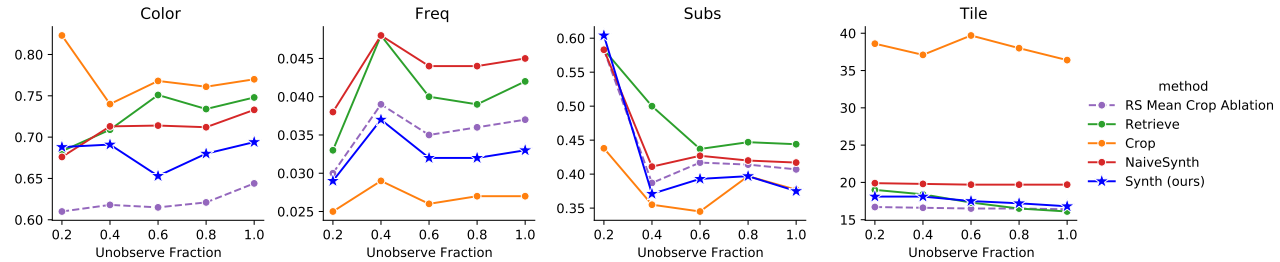


Figure 5: Evaluation on the test set across a spectrum of simulated unobserved photo fractions, reporting metrics only for *unobserved surfaces*. The RS Mean Crop ablation replaces GNN predictions with a mean crop embedding computed using all the surface crops from matching room type and surface type surfaces in the training set. `Synth` (ours) performs the best among tileable methods across different levels of unobserved photo fractions.

Table 2: Distributional similarity between generated textures and reference surface crops on the test set, measured using FID [4] score. Room types without a sufficient number of surface crops are excluded. Note that the `Crop` method has an unfair advantage as it directly uses reference crops. `Synth` performs best among all other approaches.

	`Synth` (ours)	`NaiveSynth`	`Retrieve`	`Crop`
Floor	**191.1**	238.5	289.6	65.8
Wall	**232.6**	259.2	261.0	41.8
Ceiling	**198.0**	254.4	232.7	44.1
Reception	**206.2**	253.3	262.9	47.6
Bedroom	**209.3**	250.8	247.0	31.6
Kitchen	**207.4**	256.8	256.4	49.9

than other tileable texture approaches on modelling surface type and room type specific distributions. The use of surface crops as reference crops provides `Crop` with an unfair advantage on this metric, since the crops used by `Crop` are directly sampled from the same pool of surface crops.

Table 3: Ablations showing the impact of improvements in our texture synthesis module relative to baselines. The results are reported on 4000 rectified surface crops sampled from the Rent3D dataset. The similarity metrics measure similarity between the input crop and the synthesized texture. The first two rows report values measured against input crops as baselines for comparing the TILE metric. Then, we report on ablations of our texture synthesis approach. *-NT indicates Henzler et al. [3] implemented on * color space. Δ indicates variations where median color is separated. 'SC' indicates use of the substance classifier branch. Lower values are better.

Method	VGG similarity [3]	COLOR	FREQ	SUBS	TILE
Rectified crop	**0.000**	**0.000**	**0.000**	**0.000**	52.196
Rectified crop (seam corrected)	7.439	0.098	0.019	0.279	**35.791**
RGB-NT [3]	7.295	0.724	0.054	**0.447**	**19.772**
RGB-NT + SC	**7.251**	0.832	**0.045**	0.522	19.861
Δ RGB-NT + SC	7.334	0.479	0.075	0.518	39.003
Δ HSV-NT + SC (ours)	7.375	**0.371**	0.051	0.458	21.149

E. Additional Ablations

Improvements on the texture synthesis module. Table 3 and fig. 6 show the performance of the texture synthe-

sis module using the various improvements we introduced over naively applying a texture synthesis module from prior work. The first two rows of Table 3 are naive baselines that use the input crop itself as the texture. Both of these perform the worst on the TILE metric, despite having the seam correction applied to the later case. We see that texture synthesis models trained on our stationary textures dataset do a better job on TILE. The RGB-NT approach is the neural texture approach by Henzler et al. [3]. The other approaches use the substance classifier branch during training (denoted by '+ SC'). The substance classifier encourages a structured latent space that better suits GNN propagation, despite weakening of COLOR and SUBS metrics. The VGG statistics-based similarity metric by Henzler et al. [3] gives similar numbers to all the synthesized variants. RGB-NT and RGB-NT + SC perform poorly on color similarity, as observed visually (Figure 6) and as evident by the COLOR metric values. The Δ RGB-NT + SC and Δ HSV-NT + SC methods separate the median color and feed the offset color components through the network. This drastically improves COLOR metric. As Figure 6 shows, the synthesized textures do well on substances such as wood, carpet and plastered walls but not so well on regular textures such as tiles or bricks (last two rows). This is a limitation of Henzler et al. [3]'s approach which forms the basis for our architecture. Yet, we choose to use this approach as it provided us with other desirable properties, including a compact latent embedding, and image conditioning for the textures synthesis.

Improvement on observed surfaces. The use of the textureness score to choose an embedding instead of using a mean crop embedding causes COLOR to degrade ($0.388 \rightarrow 0.453$), FREQ to improve ($0.038 \rightarrow 0.036$), SUBS to improve ($0.427 \rightarrow 0.416$) and FID to improve ($196.1 \rightarrow 185.3$). All values are reporting changes on the validation set. The above trends also apply to the test set.

Improvement on unobserved surfaces. The use of our GNN architecture for texture propagation instead of a room type and surface type conditioned mean crop embedding causes COLOR to degrade ($0.708 \rightarrow 0.747$), FREQ to improve ($0.033 \rightarrow 0.030$), SUBS to improve ($0.459 \rightarrow 0.450$) and FID to improve ($226.2 \rightarrow 196.2$). All values report changes on the validation set, with 60% unobserved photos except for FID. The above trends are consistent on both the validation and test sets across a spectrum of simulated unobserved photo fractions (see Figure 5 for plots of the test set results).

We hypothesize that mean crop emeddings outperform the textureness approach (for observed surfaces) and the GNN (for unobserved surfaces) on COLOR due to the following reason. Concatenating median RGB color to the embedding causes mean crop embeddings to synthesize textures dominated by the average color of the input crop, which is often similar to the color histogram of the medoid crop chosen as a reference. However, these mean crop embedding textures exhibit a 'washed out' appearance and score worse along other texture similarity metrics. Moreover, the FID indicates that overall (at a distributional level), textures of mean crop embeddings are of lower quality than the textureness score-based selections and GNN predictions.

Figure 6: Synthesis results from ablated versions of our improved texture synthesis module. Each synthesized texture and crop is tiled in a 2x2 configuration and textures are seam corrected using post-processing. Our texture synthesis approach produces the highest quality textures overall.

F. Additional Qualitative Examples

We provide additional qualitative comparison examples, illustrating the results of our approach and comparing them against the baselines. See Figures 7 to 10.

G. Additional User Study Analysis

The 3D renderings of the textured houses used in the user-study were generated by simulating 60% of the photos unobserved. All photos and their room assignments were shown to the users. The users were not informed about availability of a particular photo to a method. Here, we analyze the user study results per user and per house to reveal more detailed trends in user ratings comparing our approach to baselines. Figure 11 plots these results. The median of

Figure 7: Additional qualitative comparisons of results on the test set. Photos that are unobserved are indicated by dashed lines.

the preference for our approach relative to a compared baseline is indicated in each plot by a blue dashed line.

The per-user plots show a trend we did not expect: `Crop` does a relatively better job than the other two baselines. Users end up preferring `Crop` over `Synth` (ours) more than any other baseline over ours. We expected that `Crop` would produce the worst looking textures since surface crops are not tileable. However, some users mistook repeating artifacts in tiled surface crops as valid tile surface appearance. Furthermore, regularly tiled surfaces such as bathroom tiles are a common failure case for our approach and this could have also contributed.

The per house plots show that in each case only 4/20 of houses textured by our method did not receive majority preference. Two of those houses are common across all 3 experiments. These two houses are the two houses that we used in the main paper to illustrate failure cases due to erroneous semantic mask predictions and due to not accounting for illumination conditions.

Figure 8: Additional qualitative comparisons of results on the test set. Photos that are unobserved are indicated by dashed lines. Unobserved photos are uniformly sampled at random, with probability 0.6.

Figure 9: Additional qualitative comparisons of results on the test set. Photos that are unobserved are indicated by dashed lines. Unobserved photos are uniformly sampled at random, with probability 0.6.

Figure 10: Additional qualitative comparisons of results on the test set. Photos that are unobserved are indicated by dashed lines. Unobserved photos are uniformly sampled at random, with probability 0.6.

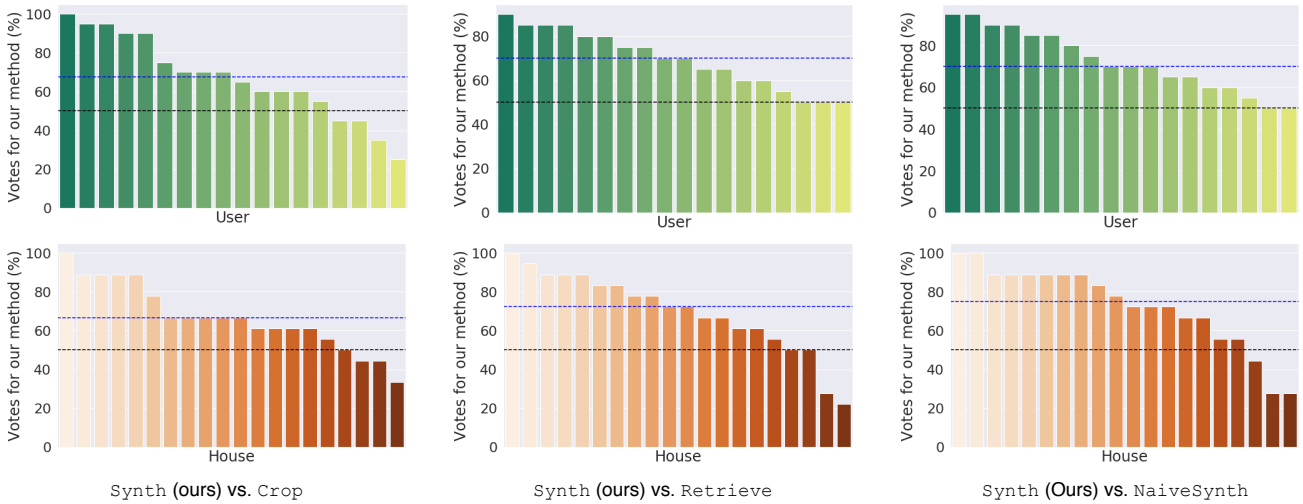


Figure 11: User study results analyzed on a per-user (green plots) and per-house (orange plots) basis. The median of preference for results produced using our approach relative to the compared baseline is indicated by the blue dashed line in each plot. The 50% preference level is indicated by the black dashed line.

References

[1] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM Trans. on Graphics (SIGGRAPH)*, 32(4), 2013. 3

[2] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. *ArXiv*, abs/1512.03012, 2015. 1, 2

[3] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Learning a neural 3D texture space from 2D exemplars. In *CVPR*, 2019. 4, 5

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017. 3, 4

[5] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Deep multi-modal image correspondence learning, 2016. 2

[6] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Raster-to-vector: Revisiting floorplan transformation. In *ICCV*, pages 2214–2222, 2017. 1