ARKitScenes Supplementary Materials

Gilad Baruch * **Zhuoyuan Chen** **Afshin Dehghan** **Tal Dimry** **Yuri Feigin** **Peter Fu**

Thomas Gebauer **Brandon Joffe** **Daniel Kurz** **Arik Schwartz** **Elad Shulman**

Apple
arkitscenes@group.apple.com

In this document we provide additional information about our dataset and ground truth, along with more details and analysis on the two downstream tasks of 3D object detection and color-guided depth upsampling.

1 Dataset

We use a custom iPadOS app for data collection with the iPad Pro. The app provides live visual feedback to the user showing a wireframe version of the on-device ARKit scene reconstruction mesh superimposed on the live camera image while capturing an RGB-D sequence. Given that the operators are not computer vision experts, the live feedback is essential for making sure the data that is collected is usable. If there is severe tracking drift in the SLAM algorithm, e.g. due to capturing an environment with lack of texture, it would be very evident in the UI. This is thanks to the efficient, low power scene reconstruction API available through the ARKit SDK. We have shown snapshots of our data collection app in Figure 1 (a) and (b). Figure 1 (c) demonstrates a sample scan pattern in a scene along with the locations of the stationary laser scanner. The operator is instructed to scan all surfaces of walls, doors, windows along with major furniture in the room.

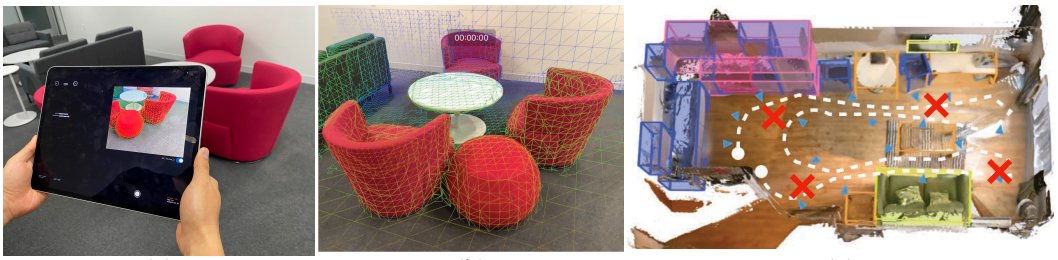

Figure 1: (a) Illustration of the iPad Pro scanning set up. (b) ARKit scene reconstruction mesh overlay to assist data collection with iPad Pro. (c) Example of one of the scan patterns captured with the iPad Pro. The red crosses show the chosen locations of the stationary laser scanner in that room.

In Figure 2 we provide detailed information about our data/label distribution along with comparison with ScanNet [1]. Our dataset covers various room types that can be seen in an indoor home including: *bedroom*, *bathroom*, *living room*, *kitchen*, *dining room*, *laundry room*, *home office*, and *recreation room*. Additionally the number of unique scenes per room type has increased significantly compared to ScanNet[1]. Moreover, we visualize the unique number of instances per

*Authors are listed in alphabetic order and contributed equally.

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks.

objects in our dataset and compare it to ScanNet. It is evident from the graph that ARKitScenes and its ground truth not only are the largest in number of unique scenes but have more than four times the number of objects labeled compared to ScanNet, even when only considering the categories that are common between the two datasets.

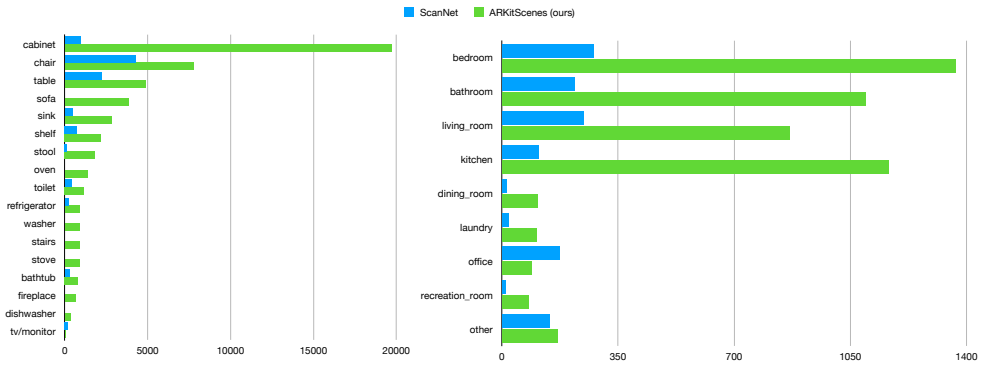

Figure 2: Distribution of ground truth object and room type instances. We offer significantly more scenes than ScanNet.

2 3D object detection

We use a customised tool to manually annotate 3D oriented bounding boxes. The labelers mark the location and orientation of the bounding boxes on the ARKit reconstructed scene mesh. Additionally our tool provides the 2D projection of the annotated 3D box into the RGB video frames of the iPad Pro. This not only increases the accuracy of labeling by providing additional information, but also reduces the confusion whenever the category label is difficult to be recognized from the mesh only. We have attached snapshots of our labeling tool.

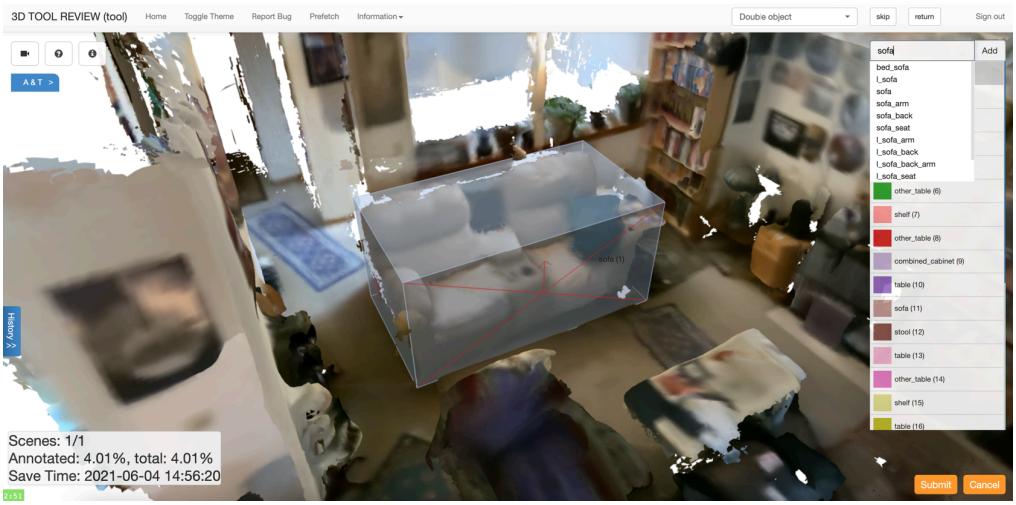

Figure 3: Screenshot showing a labeler selecting an object category.

		ℓ_1		RMSE	
		Nearest Neighbor	Bilinear	Nearest Neighbor	Bilinear
MSG	Nearest Neighbor	**0.617**	0.984	**2.279**	3.537
	Bilinear	1.076	**0.490**	4.177	**2.018**
MSPF	Nearest Neighbor	**0.810**	1.264	**2.985**	4.480
	Bilinear	1.165	**0.581**	4.985	**2.636**

Table 1: ℓ_1 and RMSE for Depth Upsampling models trained over MPI-Sintel for an upsampling factor of 8 using different interpolation methods.

	Factor	ℓ_1		RMSE	
		MPI-Sintel	ARKitScenes	MPI-Sintel	ARKitScenes
MSG	x2	1.582	**1.068**	2.650	**2.205**
	x4	1.578	**1.046**	2.643	**2.187**
	x8	1.598	**1.084**	2.709	**2.246**
MSPF	x2	1.582	**0.962**	2.671	**2.161**
	x4	1.555	**0.953**	2.627	**2.142**
	x8	1.559	**0.953**	2.708	**2.139**

Table 2: ℓ_1 and RMSE for Depth Upsampling trained over MPI-Sintel and ARKitScenes train set, and evaluated over ARKitScenes validation set.

3 Color-guided depth upsampling

3.1 Comparison to existing datasets

In order to better quantify the contribution of ARKitScenes to the field of color-guided depth upsampling, we designed two experiments: The first will prove that existing datasets are not applicable to real-world scenarios by showing their dependency on the method used to generate the low resolution (LR) training and validation images. The second experiment is showing the performance of models trained on existing datasets and evaluated over ARKitScenes.

For those experiments we used the MSG [2] and MSPF [3] architectures, similar to what we used in the main paper. MPI-Sintel [4] was selected as the dataset for comparison, as it is the most common dataset for depth upsampling to date. Disparity in pixels is the native representation of data in MPI-Sintel, so we converted ARKitScenes data to disparity to facilitate a fair comparison.

3.1.1 Existing datasets dependence on LR generation method

In this experiment we show that existing datasets in which the low resolution (LR) image is generated by downscaling the high resolution (HR) image produce models which overfit to the method used to generate the LR images.

We trained models for an upsampling factor of 8, once with the LR images in the trainin set generated using *Nearest Neighbor Interpolation*, and once using *Bilinear Interpolation*. We then evaluated those models on two versions of the validation set using the same two interpolation methods. Results can be found in Table 1. It is clear that evaluating over images generated using an interpolation method different from the one used to generate the training dataset degrades the results significantly: both ℓ_1 and RMSE are reduced by 33% to 54%. These results lead to the conclusion that models that were trained using artificial downscaling of HR images cannot generalize well to the real world in which low cost sensors, such as Apple's LiDAR scanner, produce noisy LR images.

3.1.2 Training over existing datasets

To quantify the difference between ARKitScenes and existing datasets, we trained reference models on ARKitScenes, while other models were trained using MPI-Sintel [4]. The results are shown in Table 2. One can observe that models trained on MPI-Sintel perform poorly on ARKitScenes. On the other hand when we use ARKitScenes for training, we can improve the results by 30% to 46% in ℓ_1, and 16% to 26% in RMSE.

3.2 Adaptations of losses for ARKitScenes

The ground truth depth maps are re-projections of the measurements of a laser scanner taken from different viewpoints, occlusions will cause lack of information in some regions. We use zeros as the depth value in such regions. In this section we will detail the adaptations required to the losses being used in the literature due to this effect.

3.2.1 ℓ_1 loss

The ℓ_1 loss is calculated per-pixel as the absolute difference between the prediction and the ground truth: $|p_{prediction} - p_{ground\_truth}|$. In order to perform well on ARKitScenes, all that is required is ignoring pixels at which the ground truth is 0 (denoting no-depth).

3.2.2 Structural Similarity Index Measure (SSIM) loss

SSIM is a metric for evaluating the *perceived* quality of an image by measuring the similarity of the prediction to a reference image while taking into account perception phenomena such as luminance and contrast masking.

SSIM requires the full ground truth image without masks for structure calculation, as masks will inherently change structural information. As such, this loss cannot be used over ARKitScenes ground truth depth maps which contains masked pixels due to occlusions.

3.2.3 Edge loss

A loss suggested by [3] to ensure the sharpness of edges is defined as:

$$L_{edge}(\Theta) = \frac{1}{N}||sobel(F(D^{LR}, I^{HR}; \Theta)) - sobel(D^{HR})||_1 \tag{1}$$

where D^{LR} is the low resolution depth map, I^{HR} is the guiding high resolution color image and D^{HR} is the high resolution ground truth depth map. *sobel* is the 5×5 filter that computes smooth gradients of an image. As explained before, this operator cannot be applied on the ground truth of ARKitScenes because of the occlusions that will cause hallucinated edges around regions without information. In order to overcome this issue one might naïvely mask out regions with no information from the gradients image. However, such an approach will yield poor results because the operator is convolving information from around each pixel. To combat this the no-depth mask should be dilated in order to prevent leakage of wrong values to the gradient. However, once using sufficient dilation, the information around real edges is usually lost.

To help reduce the impact of this problem, we replaced the edge loss originally used in [3] with the edge loss suggested by [5], defined as an ℓ_1 penalty on differences in log-depth gradients at different scales between the predicted and ground truth depth map:

$$L_{edge} = \frac{1}{n}\sum_k \sum_i (|\nabla_x R_i^k| + |\nabla_y R_i^k|) \tag{2}$$

where R_i^k is the value of the log-depth difference map at position i and scale k. To adapt this loss to our ground truth all that is needed is to discard positions at which there is no depth information.

3.3 Experiment settings

Depth upsampling methods are usually tested using upsampling factors which are a factor of 2 on each axis. In our dataset the resolution of the LR image is 256×192 and the resolution of the HR image is 1920×1440. When possible we preferred to keep the LR image as is and downscale the ground truth. For the case of x2 and x4 upsampling we downscaled the HR images to 512×384 and 1024×768 respectively. For x8 upsampling we used the original HR image and downscaled the LR image to 240×180. After resizing, we rotate the images to be in portrait instead of landscape (e.g. 192×256 instead of 256×192) when needed to reflect the capturing orientation. Due to the high resolution of the images, fitting all the images in the batch in GPU memory is prohibitively large. Hence, we randomly cropped the HR depth maps in the training set to patches of size 512×512 for scale factors 4, 8 and 256×256 for scale factor 2.

For color image interpolation we used linear interpolation, while depth maps were interpolated using nearest-neighbor interpolation to avoid smooth edges and artifacts.

In our implementation we followed the settings used in [3] to employ ADAM to optimize the parameters of the network. We start with a learning rate of 10^{-4} with linear decay in the entire training procedure. We train the proposed networks with a batch size of 16 for 100k iterations to convergence. We augment each patch by randomly flipping the horizontal axis.

3.4 Depth upsampling examples

More examples for color-guided depth upsampling can be found in Figure 8. The edges produced by the data-driven MSPF are sharper than other methods, except for the FGI case in which the transition is sharp but the edge itself is jagged and in low resolution.

References

[1] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[2] Tak-Wai Hui, Chen Change Loy, , and Xiaoou Tang. Depth map super-resolution by deep multi-scale guidance. In *Proc. European Conference on Computer Vision (ECCV)*, pages 353–369, 2016.

[3] Chuhua Xian, Kun Qian, Zitian Zhang, and Charlie CL Wang. Multi-scale progressive fusion learning for depth map super-resolution. *arXiv preprint arXiv:2011.11865*, 2020.

[4] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *Proc. European Conference on Computer Vision (ECCV)*, pages 611–625. Springer, 2012.

[5] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

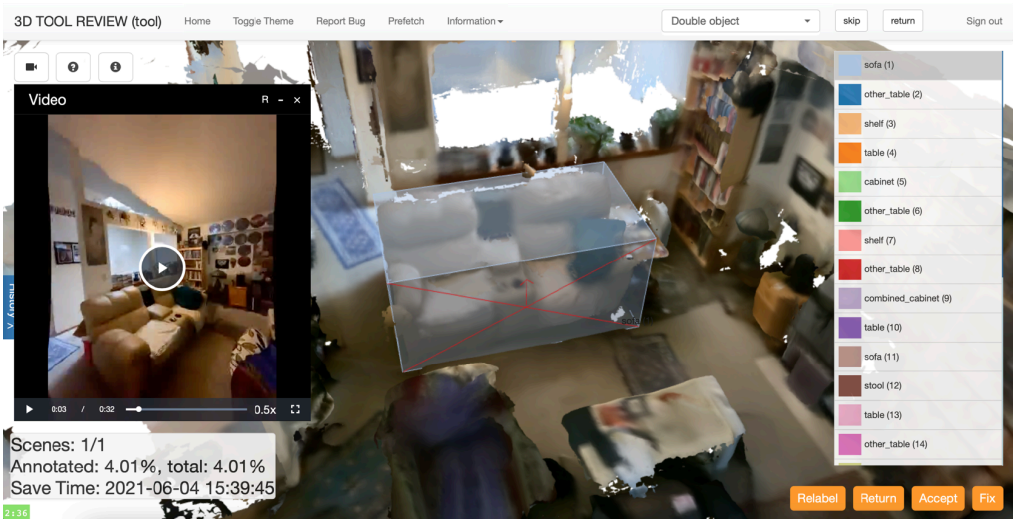

Figure 4: Our annotation platform displays the original video and the ARKit scene reconstruction color mesh side by side to provide more context for the labelers, thus leading to more accurate labeling.

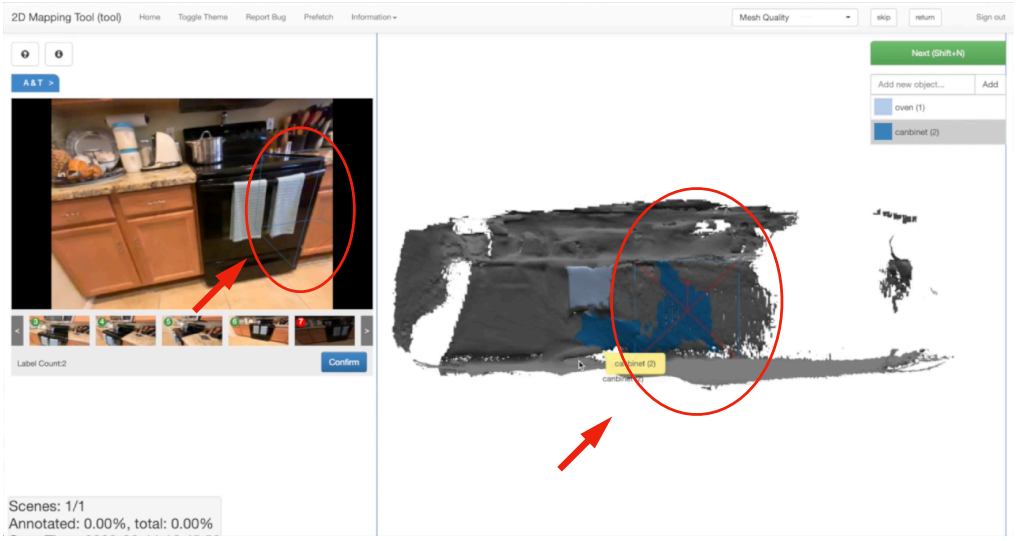

Figure 5: Our labeling tool enables real-time 2D projection of 3D bounding boxes into video frames to facilitate accurate annotation.

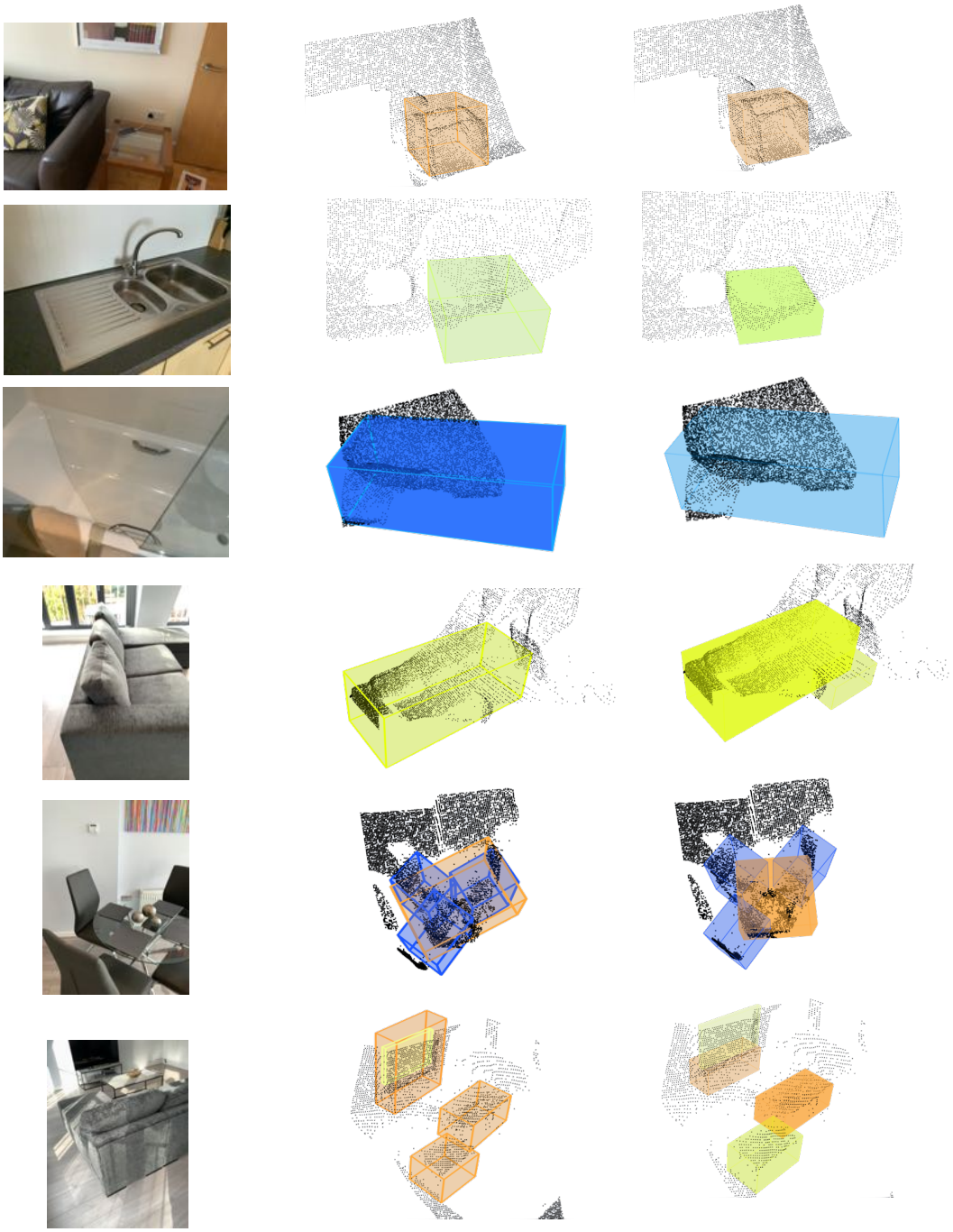

Figure 6: Examples of RGB-D frames, predictions (DT) and ground truth bounding boxes (GT) for Single-Frame settings.

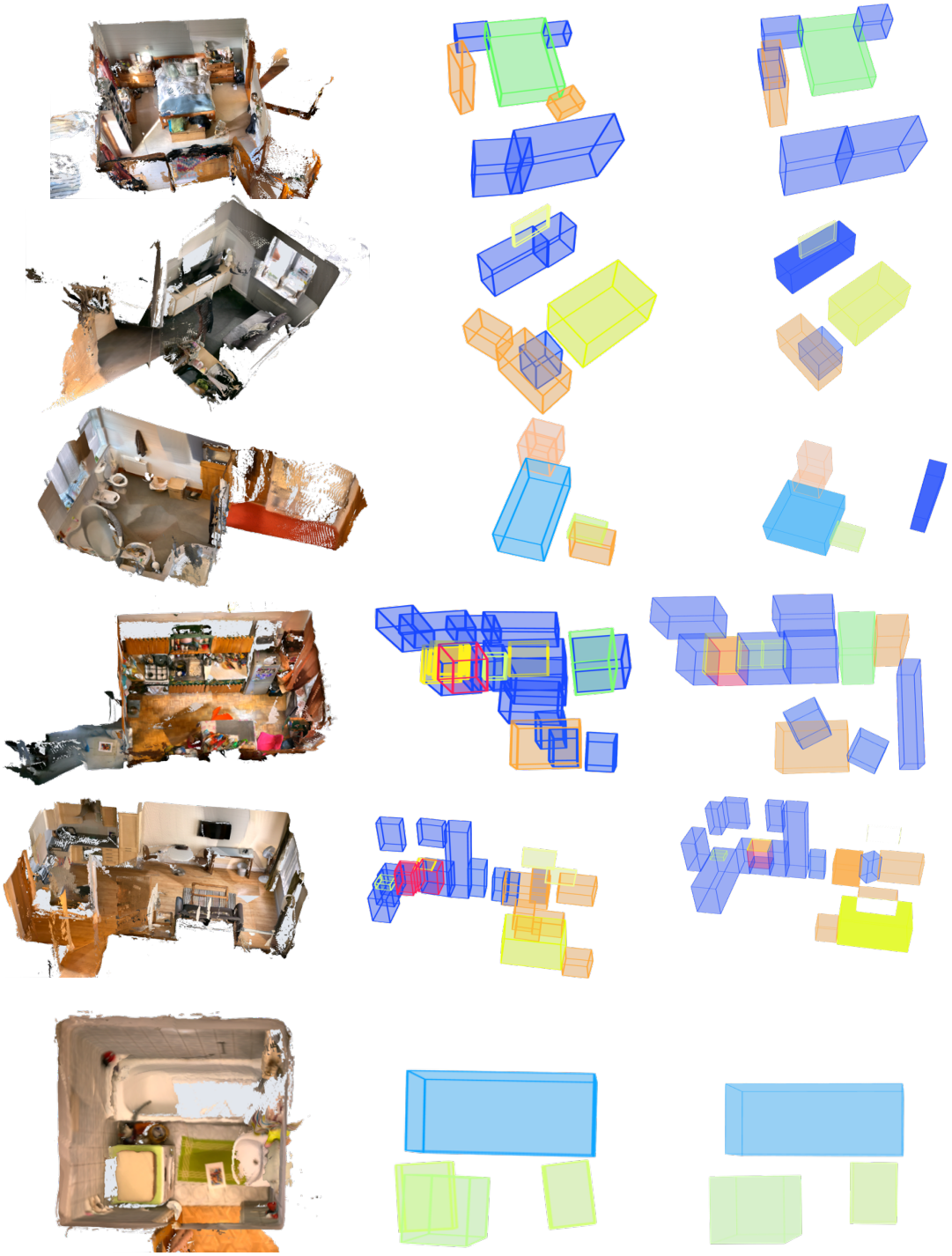

Figure 7: Examples of ARKit scene reconstruction meshes, prediction (DT) and ground truth bounding boxes (GT) for Whole-Scene settings.

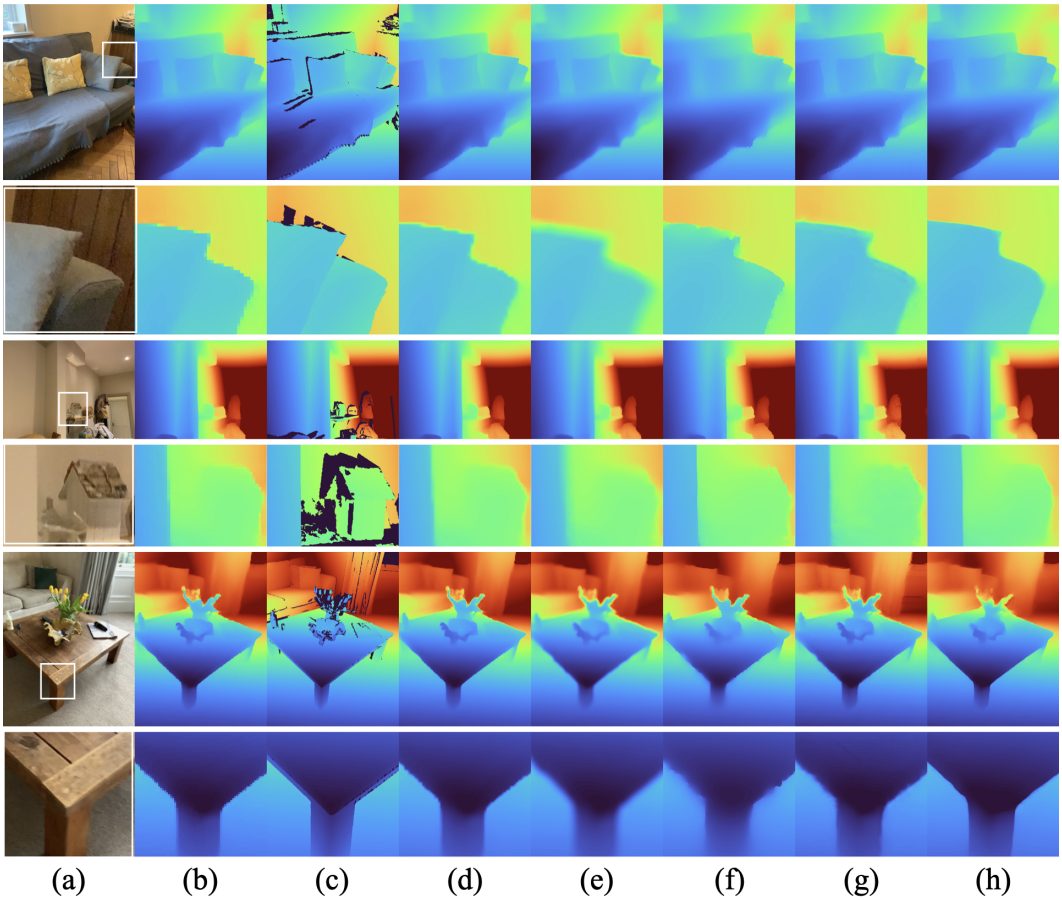

Figure 8: Examples of upsampling results using different methods with an upsampling factor of 8. Each example is depicted in two rows; first for the entire image followed by a row with a crop of an interesting part of the image. The images denote: (a) HR color, (b) LR ARKit depth map,(c) HR ground truth and the results of the different upsampling methods: (d) Bilinear, (e) JBU, (f) FGI, (g) MSG and (h) MSPF.