Scene understanding is an active research area. Commercial depth sensors such as Kinect, have enabled the release of several RGB-D datasets over the past few years which spawned novel methods in scene understanding. More recently with the launch of LiDAR sensor in Apple’s iPads and iPhones, high quality RGB-D data is accessible to millions of people on a device they commonly use. This opens a whole new era in scene understanding for the Computer Vision community as well as the developers. The fundamental research in scene understanding together with the advances in machine learning can now impact people’s everyday experiences. However transforming these fundamental scene understanding methods to real world experiences, requires additional innovation and development. In this paper we introduce ARKitScenes. ARKitScenes is not only the first RGB-D dataset that is captured with now widely available depth sensor, but also is the largest indoor scene understanding data ever collected. In addition to the raw and processed data, ARKitScenes includes high resolution depth maps captured using a stationary laser scanner, as well as manually labeled 3D oriented bounding boxes for a large taxonomy of furniture. We further analyze the usefulness of the data for two downstream tasks: 3D object detection and RGB-D guided upsampling. We demonstrate that our dataset can help push the boundaries of existing state-of-the-art methods and introduce new challenges that better represent real world scenarios.
hardware used to collect depth data, such as the one used to collect SunRGBD [15] or ScanNet [14], are different from the hardware used by people nowadays. The lack of diversity in data and the gap in the depth sensing technology brings challenges in making the innovative research of the last decade practical for day to day use.

Recently, Apple released iPads and iPhones equipped with the LiDAR scanner [17]. It unleashed a new era in availability and accessibility of depth sensors. This work provides the first large-scale dataset that is captured with Apple’s LiDAR scanner using handheld devices. It helps bridge the domain gap between existing datasets and widely available mobile depth sensors, and is the largest RGB-D dataset in terms of number of sequences and scene diversity collected in people's homes. We named this dataset ARKitScenes.

ARKitScenes consist of 5,048 RGB-D sequences which is more than three times the size of the current largest available indoor dataset [14]. These videos include 1,661 unique scenes. Additionally we provide ground truth camera poses as well as the surface reconstruction for all the videos. A comparison of ARKitScenes with some of the existing datasets is shown in Table 1. In addition to the raw and processed data above, we provide high quality ground truth and demonstrate its usability in two downstream supervised learning tasks: 3D object detection and depth up sampling. For the 3D object detection task, ARKitScenes provides the largest RGB-D dataset labeled with oriented bounding boxes for 17 room-defining categories. ARKitScenes provides high-resolution ground truth that is captured with a professional stationary laser scanner (Faro Focus S70). We include a description of a technique used to register the high quality laser scans with mobile phone RGB-D frames captured with an iPad Pro. To our best knowledge, this is the first dataset that provides high quality ground truth depth data registered to a widely available depth sensor. Finally, we evaluate the performance of state-of-the-art methods when trained and evaluated on ARKitScenes. Our experiments unveils the limitation of the current state-of-the-art and highlights the challenges of existing methods in generalizing to realistic scenarios. In summary the contributions of this paper are as follows:

- We present ARKitScenes, the first RGB-D dataset captured with the widely available Apple LiDAR scanner. Along with the raw data we provide the camera pose and surface reconstruction for each scene.

- ARKitScenes is the largest indoor 3D dataset consisting of 5,048 captures of 1,661 unique scenes.

- We provide high quality ground truth of (a) registered RGB-D frames and (b) oriented bounding boxes of room defining objects.

- We demonstrate the effectiveness of the dataset in advancing the state of the art methods, while highlighting the limitations of current methods and datasets in generalizing to realistic scenarios.

We expect ARKitScenes to stimulate the development of novel algorithms and spawn new research directions. Furthermore, we call for an evaluation and comparison of future work on ARKitScenes as it represents a diverse set of homes in the wild. And finally, we hope ARKitScenes bridges the gap between innovation and usability by the general public as it provides data captured with a sensor similar to the one people carry in their pockets.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>3DOD Labels</th>
<th>HR #Frames</th>
<th>HR</th>
<th>LR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI-Sintel</td>
<td>35 Scenes</td>
<td>-</td>
<td>1,628</td>
<td>1024×436</td>
<td>-</td>
</tr>
<tr>
<td>Middleburry</td>
<td>-</td>
<td>-</td>
<td>34</td>
<td>432×381 to</td>
<td>2964×2000</td>
</tr>
<tr>
<td>NYU v2</td>
<td>464 Scans</td>
<td>1,449 frames</td>
<td>-</td>
<td>-</td>
<td>640×480</td>
</tr>
<tr>
<td>SUN RGB-D</td>
<td>10K frames</td>
<td>10k frames</td>
<td>-</td>
<td>-</td>
<td>512×424 to</td>
</tr>
<tr>
<td>SceneNN</td>
<td>100 Scans</td>
<td>100 scans</td>
<td>-</td>
<td>-</td>
<td>640×480</td>
</tr>
<tr>
<td>ScanNet</td>
<td>707 Venues, 1,513 Scans</td>
<td>1,513 scans</td>
<td>-</td>
<td>-</td>
<td>640×480</td>
</tr>
<tr>
<td>ARKitScenes</td>
<td>1,665 venues, 5,048 Scans</td>
<td>5048 scans</td>
<td>400K</td>
<td>1920×1440 Laser Scanner</td>
<td>256×192 ARKit Depth</td>
</tr>
</tbody>
</table>

Table 1: Overview of RGB-D datasets and their ground truth asset comparison with ARKitScenes. HR and LR represent High Resolution and Low Resolution respectively.

2 Related work

3D Scene Understanding. Availability of large-scale datasets such as ImageNet [18] stimulated research, especially since supervised deep learning techniques re-gained popularity. In the context of scene understanding and 3D point clouds, several datasets were released in the past few years [16,14]. These datasets have enabled a series of research investigations in various areas including semantic segmentation, object detection, room layout estimation, depth estimation and more. For outdoor scene understanding, several large scale datasets with a variety of scenes in real scenarios were released that have powered deep learning algorithms [19,13,20,21]. However, when it comes to indoor scene understanding we are only limited to a few datasets for deep learning investigations [14,15,16,22]. Outdoor and indoor datasets have very different characteristics, both caused by the size of the space and the type of sensors that are used to collect those datasets. Because of that, the methods designed for one will not necessarily perform the same in the other.

Indoor Scene Understanding. NYU v2 [16] is one of the earliest RGB-D datasets focusing on indoor scenes. It is composed of 464 scenes from 3 cities captured with a Kinect device. It also includes 1,449 densely labeled pairs of aligned RGB and depth maps annotated with 2D polygons. Sun RGB-D [15] further expands on previous work by introducing a dataset with over 10,000 RGB-D frames along 2D polygon and 3D bounding box labels. The labels are provided at frame level and not scene level, lacking view-point diversity. There are several other datasets which have focused on long indoor RGB-D captures [9,14,23,24] to address the view-point diversity. Among these datasets ScanNet [14] is the largest and closest to ours in terms of scene diversity and assets. ScanNet provides 1,513 scans of 707 unique scenes along with dense 3D labels and CAD models. Our dataset on the other hand is 3× the size of ScanNet, captured with Apple's LiDAR scanner instead of Kinect, and it provides 3D oriented bounding boxes and depth ground truth which were missing in [14].

3D Object Detection is a computer vision task that has gained a lot of popularity in recent years [25,26,27,28,29,30,31,4,5,32,7]. The majority of the published techniques focus on outdoor environments [4,5] mainly in the context of autonomous driving, where diverse datasets such as [19,13] exists. Most of these techniques make assumptions in the algorithm (e.g Bird's Eye View projection) that do not generalize well to indoor scenes. For indoor 3D object detection the number of datasets is limited. SunRGB-D and ScanNet are the two most commonly used ones. The former lacks scene level labels and the latter lacks oriented 3D bounding box labels. More recent datasets such as [34] provide a variety of objects with 3D labels but do not provide depth sensor data. ARKitScenes provides the largest set of 3D oriented bounding boxes for a set of 17 room-defining object categories that addresses the gaps of previous indoor datasets.
Color-Guided Depth Upsampling is the task of generating a high resolution (HR) depth map by using a high-resolution color image as guidance for the upsampling of a low resolution (LR) depth map \[35, 36, 37\]. HR depth maps are essential for many depth use cases which require high frequency depth information. Prior works are using datasets of a few images with high resolution ground truth, such as 34 images from Middlebury \[38, 39, 40\], or 58 synthetic images from MPI-Sintel \[41\], or using the low resolution depth sensors as HR ground truth \[15, 16\], and downsampling it even further in order to obtain the LR image. Table 1 compares those datasets and their properties. ARKitScenes is unique since it does not use the ground truth image as the source to generate the LR image by simple downsampling, instead it is providing LR depth maps captured with a consumer grade sensor and registered ground truth depth maps captured with a professional stationary laser scanner. As a result upsampling methods trained with ARKitScenes are expected to generalize better to real-world scenarios as will be demonstrated below.

The rest of this paper is organized as follows. First, in Section 3, we introduce our data collection protocol as well as details around hardware and software used to capture the data. Moreover we cover the details around how we gather ARKitScenes ground truth. Next, in Section 4, we explore ARKitScenes for two downstream tasks of 3D object detection and depth upsampling. Finally, in Section 5, we summarize our findings and propose some future work.

3 ARKitScenes Dataset

In this section, we describe the steps we pursued to acquire this dataset from collecting raw data in real-world homes, our data collection app, to fully-automatic spatial registration of the collected video sequences with highly accurate high-resolution stationary laser scans as well as manual annotation of 3D bounding box labels in the dataset.

3.1 Raw Data Acquisition

We used two main devices for data collection: The 2020 iPad Pro and Faro Focus S70. The 2020 iPad Pro is used to collect various sensor outputs such as IMU, RGB (for both wide and super-wide cameras) as well as the dense depth map via ARKit. We use the official ARKit SDK \[2\] to collect such information. Our data collection app runs ARKit motion tracking and surface reconstruction during the capture. This is to provide live feedback to the operators, who are not computer vision experts, on tracking robustness and reconstruction quality. In addition to the iPad Pro we utilized a Faro Focus S70 stationary laser scanner to collect high-resolution XYZRGB point clouds of the environment.

For data collection locations, we use real-world homes which we rent for a full day. The home owners consented to this data being released publicly to facilitate research and development of indoor scene understanding. The operators was instructed to remove any personally identifiable information prior to starting the captures. Data is collected in Europe and three major cities, London, Warsaw and New Castle. To increase the indoor scene diversity and coverage we took two criteria into account when selecting a home for data collection, the socioeconomic status (SES) of the household as well as the location of the house in the city. The houses in our dataset are selected from rural, suburban and urban location in each of the aforementioned cities. Additionally we included houses from all three tiers of low, medium and high SES levels.

\[2\]https://developer.apple.com/documentation/arkit
After selecting the house for data collection, we divide each house into multiple scenes (in most cases, each scene covers one room), and perform the following steps. First, we use a Faro Focus S70 stationary laser scanner on a tripod to collect highly accurate XYZRGB point clouds of the environment. Tripod locations are chosen to maximize surface coverage, and on average we collected four Faro scans per room to ensure good coverage. Second, we record up to three handheld videos performing full scans for each room using the iPad Pro. Each scan follows a different scan pattern and captures the ceiling, floors, walls and room defining objects. With an on-device reconstructed mesh overlaid on the camera stream in the data collection app, we ensure the objects in the room are well covered.

Throughout the collection of all data, we attempt to keep the environment completely static, i.e. we make sure no objects move or change their appearance. However, since data collection of a venue takes an average of six hours and many venues are lit by sunlight, the lighting situation can change during that time resulting in potentially inconsistent illumination between the different sequences and scans.

### 3.2 Ground truth generation

**Ground truth poses and depth maps.** After having collected all data, in a one-time offline step, we first spatially register all XYZRGB point clouds from the stationary laser scanner into a common coordinate system using the proprietary software Faro Scene. For most environments, this tool works fully automatically and estimates a 6DoF rigid body transformation for each scan transforming it into a common venue coordinate system. Note that a venue (usually a house or apartment) can comprise multiple unique scenes. After this step, and throughout the rest of this paper, the XYZRGB point clouds are always assumed to be expressed in common venue coordinates.

Our approach to estimate the ground truth 6DoF pose of the iPad Pro’s RGB cameras with respect to the venue coordinate system requires the generation of synthetic views of our scan of the venue. Rendering these XYZRGB point clouds from novel viewpoints poses unique challenges. In particular, we require that far geometry is correctly occluded by near geometry and that geometry is discarded for which a direct line-of-sight from the novel view point cannot be guaranteed (e.g. they might be occluded by surfaces that were not captured in the scan). Obviously, naively rasterizing the scans as unstructured point clouds would violate both of these requirements. In the following we describe a method that employs standard rendering techniques to efficiently rasterize multiple point clouds with accurate handling of occlusion and reasoning about line-of-sight.

First, for each scanned 3D point cloud a triangulation is found by reducing it to two dimensions via stereographic projection with respect to the laser scanner’s nodal point and computing a 2D

---

3An example of such scan patterns and our app UI is shown in our supplementary materials.
Figure 4: Prediction (middle column) and ground truth boxes (right column) for Single-Frame and Whole-Scene detection.

Delaunay triangulation. When applied to the 3D point cloud this triangulation is forming a watertight mesh. We further project the RGB color information of each point into an equirectangular texture map and assign corresponding texture coordinates to each vertex in the mesh, resulting in a watertight textured triangle mesh for each stationary laser scan.

The triangles of this mesh are then split into two sets by applying a threshold on the angle between the triangle normal and the ray from the triangle center to the laser scanner nodal point. When this angle exceeds a threshold the triangle is considered to manifest a discontinuity and will be used as occlusion geometry; otherwise it will be used as foreground geometry, see figure 3 (b) for an example visualization.

Finally, the two sets of triangles are rendered in separate passes using OpenGL, both writing to the depth buffer, while only front-facing triangles of the foreground geometry write a nonzero value to the stencil buffer. In all other cases the stencil buffer is cleared. As a result only fragments with unobstructed line-of-sight towards front-facing foreground geometry will have a nonzero stencil value, hence the stencil buffer can be used to mask pixels in the rasterized output. To create a joint rendering of multiple scans each scan is rendered separately and the renderings are merged in screen space using the individual depth and stencil buffers: starting from the rasterization result of the first scan, the rasterization result of an additional scan takes precedence on pixels with nonzero stencil and smaller depth value. After repeating this process for every scan the rasterization of a synthetic view is complete.

In summary this method can be used to rasterize depth maps and RGB images of one or multiple stationary laser scans from arbitrary viewpoints.

To be used as ground truth, the next objective is to determine the 6DoF pose of the handheld iPad Pro’s cameras with respect to the venue coordinate system. To this end we extract local image features and descriptors in keyframes of the camera sequence and store them as query features. Using the presented rasterization method we then create renderings of the LiDAR scans, in which we detect and describe the same kind of local image features and store them as reference features together with their 3D locations, which can be looked up in the rasterized depth maps. We then match each query feature with the most similar reference feature leading to a set of 2D-3D correspondences for each keyframe. We use RANSAC [42] and PnP to estimate initial camera poses using these sparse correspondences. To improve over these initial estimates we jointly solve for the camera poses of all keyframes that minimize a dense photometric error metric. For a single keyframe the metric optimizes photometric consistency between (1) the keyframe image and rendered views of the LiDAR scans and (2) the keyframe image and projected views of neighboring keyframes that share visibility of the same parts of the LiDAR surface geometry. This registration is performed once offline per video sequence.

When the refined ground truth poses are determined we render camera frame-aligned ground truth depth maps encoding per-pixel orthographic depth. This dense ground truth depth map, along with the per-keyframe ground truth pose are provided as part of the dataset. Figure 5 visualizes the results for a set of keyframes from the dataset.
3D Object Bounding Boxes. We use a custom tool to manually annotate 3D oriented bounding boxes for 17 categories of room-defining furniture. The annotation happens on the reconstructed colored mesh of the whole scene. Additionally, our labeling tool allows labelers to see real-time projections of 3D bounding boxes onto video frames to facilitate accurate annotation. To further enrich our annotation, each bounding box is labeled with an additional attribute called completeness that reflects how well the object was scanned. Completeness can take any of the labels: high (50%-100%), or low (0-50%). We provide additional information about our labeling tool and dataset distribution, along with completeness study and its impact on 3D Object Detection algorithm in supplementary material.

4 Tasks and Benchmarks

To evaluate the accuracy machine learning models can achieve when trained on ARKitScenes, we chose two computer vision tasks and trained state-of-the-art models using our dataset.

Train/Validation/Test split. We split the video sequences of our ARKitScenes into 80% for training, 10% for validation and the remaining 10% are a held-out test set that we do not release. The training and validations set include the 5,048 sequences which we intend to release upon acceptance of this paper. The split is shared across tasks in both 3D object detection and depth upsampling. For binning scans and sequences, we utilized the venue ID which is a unique ID assigned to each asset. This is to make sure that all scans belonging to the same venue (i.e house) appear in the same split.

4.1 3D object detection

Problem Details. As a fundamental task in computer vision, the goal of 3D object detection is to localize and recognize objects in a 3D scene. In ARKitScenes, we focus on two setups for our 3D object detection: single-frame based and whole-scan based. Given a video sequence of RGBD scan, the former targets detecting objects in each single frame, while the latter we detect objects from the whole 3D scene.

Ground Truth Processing. Our ground truth bounding boxes are labeled on each scan, we can naturally use all of them as our ground truth for whole-scan based 3D object detection. However, for single-frame, we need to pre-process our ground truth data for the single-frame object detection task. In outdoor object detection benchmarks such as Kitti [19], a common practice is to keep a bounding box if at least 5 corners are in range. For indoor scenario, the case is more complicated due to occlusion. It is common that a chair in view is occluded by a dining table at front. Accordingly, we apply two criteria for box filtering for single-frame detection. First, a box is removed if there are at most 4 corners in our camera frustum. Second, a box is removed if its completeness is lower than 50% in the current frame. We define completeness by the ratio of point number of current frame and the whole scan. Given a bounding box, if there are n points in current frame and N points in the accumulated whole scan, we accept the box if n > 0.5N. In some scenarios, an object appearing in many frames will have N >> 2n and its bounding box will always be removed. We conquer this problem by density standarization: we divide our 3D space into grids of size 5cm × 5cm × 5cm and keep at most one point in each grid. If a box is removed, we removed the point cloud inside the box to avoid confusion and discourage hallucination. After this filtering step we are left with over four million frames. Using all four million data for training is time consuming for modern neural network. To speedup training and evaluation, we randomly

<table>
<thead>
<tr>
<th>Categories</th>
<th>Cabinet</th>
<th>Refrigerator</th>
<th>Shelf</th>
<th>Stove</th>
<th>Bed</th>
<th>Sink</th>
<th>Washer</th>
<th>Toilet</th>
<th>Bathtub</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Frame</td>
<td>0.785</td>
<td>0.624</td>
<td>0.333</td>
<td>0.144</td>
<td>0.933</td>
<td>0.781</td>
<td>0.637</td>
<td>0.946</td>
<td>0.962</td>
</tr>
<tr>
<td>Whole-Scan</td>
<td>0.336</td>
<td>0.501</td>
<td>0.138</td>
<td>0.005</td>
<td>0.854</td>
<td>0.331</td>
<td>0.338</td>
<td>0.799</td>
<td>0.925</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Categories</th>
<th>Oven</th>
<th>Dishwasher</th>
<th>Fireplace</th>
<th>Stool</th>
<th>Chair</th>
<th>Table</th>
<th>TV/Monitor</th>
<th>Sofa</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Frame</td>
<td>0.553</td>
<td>0.354</td>
<td>0.737</td>
<td>0.312</td>
<td>0.805</td>
<td>0.679</td>
<td>0.360</td>
<td>0.919</td>
<td>0.636</td>
</tr>
<tr>
<td>Whole-Scan</td>
<td>0.163</td>
<td>0.029</td>
<td>0.301</td>
<td>0.039</td>
<td>0.177</td>
<td>0.263</td>
<td>0.060</td>
<td>0.080</td>
<td>0.341</td>
</tr>
</tbody>
</table>

Table 2: Evaluation of VoteNet[4] on single-frame and whole-scan object detection tasks on our ARKitScenes dataset.
sample a subset of 10% frames from each scan, which contains about 400K frames for training and 40K for validation.

**Models and Training Details.** Building on the PointNet++ backbone and Hough voting modules, VoteNet [4] achieves state-of-the-art performance on indoor scenarios. Along this line, imvotenet [43], MLCVNet [28] and H3DNet [43] further improves the VoteNet model by leveraging RGB information [5], Graph-Neural Network [28] or extra geometric primitive prediction. Without generality, we chose the original VoteNet model for our evaluation.

We followed the original design of VoteNet model [4]: the backbone network is a PointNet++ with several set-abstraction layers and feature propagation (upsampling) layers with skip connections, which outputs a subset of the input points with XYZ and an enriched \( C \)-dimensional feature vector. The results are \( M \) seed points of dimension \((3+C)\). Each seed point generates one vote. Each seed goes through a Hough voting module with supervision to guide each foreground point to vote to its bounding box center. Finally a last proposal module aggregates the votes with a shared PointNet to predict center, size and category of each bounding box. As in [4], we use ADAM optimizer to train our model for 200 epochs, with learning rate 0.001 and decay rate 0.1 at the 80-th and 120-th epoch. We augment our data with rotation, scaling and translation.

**Benchmark.** As a baseline evaluation, we show the performance of object detection on single-frame and whole-scan in Table 2. VoteNet achieves mAP (mean average precision) of 0.636 for single frame detection and 0.341 for whole-scan detection. Specifically, VoteNet generally performs better on large objects such as cabinet, refrigerator and bed, and struggles on small objects such as stove, dishwasher and stool. Figure 4 shows qualitative results of VoteNet on ARKitScenes. For more results and additional experiments please refer to supplementary material.

### 4.2 Color-guided depth upsampling

**Problem Details.** Depth upsampling is a common approach used to enhance low resolution (LR) depth maps to a high resolution (HR), higher fidelity depth map using an HR color image as guidance. HR accurate depth maps are imperative for downstream tasks such as 3D reconstruction, augmented reality, and photography. All of these require high frequency depth information which is often lost at low resolution.

Initial classical approaches to color-guided depth upsampling include both optimization and filtering based methods [36,37]. These approaches performed relatively well but are often hand-crafted and lack the ability to capture global structure and context. More recently, a data-driven approach using deep neural networks helps overcome some of these challenges. One of the prominent works in this area is *Multi-Scale Guided Networks* (MSG) [35], an encoder-decoder network extracting features at different resolutions from the guiding image in the encoder branch, and concatenating it in the corresponding resolution of the decoder branch of the depth map. The network is trained to learn the differential correction of the naive upsampling. The current state-of-the-art in the field of guided depth upsampling is *Multi-Scale Progressive Fusion* (MSPF) [44], where the authors suggest the use of two different encoder branches, one for depth and one for color, along with a reconstruction branch that applies fusion blocks to restore the HR depth map.

**ARKiTScenes Adaptations.** As mentioned in Section 2, prior works on depth upsampling were demonstrated over LR depth maps that were generated by down-sampling ground truth HR depth maps, sometimes with the addition of artificial noise. This inherently makes these datasets easier for processing but limits the evaluation to non-realistic scenarios in which the low resolution sensor is only suffering from synthetic artifacts not necessarily representative of real-world scenarios. However, ARKitScenes brings a more realistic challenge of upsampling a low resolution depth map as it is captured on a mobile device that has artifacts inherent to active sensing. Hence the challenge becomes twofold: depth upsampling and depth artifacts correction.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Bilinear</th>
<th>JBU</th>
<th>FGI</th>
<th>MSG</th>
<th>MSPF</th>
<th>Bilinear</th>
<th>JBU</th>
<th>FGI</th>
<th>MSG</th>
<th>MSPF</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x2)</td>
<td>0.0322</td>
<td>0.0328</td>
<td>0.0324</td>
<td>0.0242</td>
<td><strong>0.0220</strong></td>
<td>0.0640</td>
<td>0.0639</td>
<td>0.0652</td>
<td>0.0590</td>
<td><strong>0.588</strong></td>
<td></td>
</tr>
<tr>
<td>(x4)</td>
<td>0.0321</td>
<td>0.0328</td>
<td>0.0326</td>
<td>0.0237</td>
<td><strong>0.0214</strong></td>
<td>0.0639</td>
<td>0.0637</td>
<td>0.0649</td>
<td>0.0584</td>
<td><strong>0.581</strong></td>
<td></td>
</tr>
<tr>
<td>(x8)</td>
<td>0.0326</td>
<td>0.0322</td>
<td>0.0332</td>
<td>0.0253</td>
<td><strong>0.0217</strong></td>
<td>0.0662</td>
<td>0.0650</td>
<td>0.0658</td>
<td>0.0592</td>
<td><strong>0.583</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: \(\ell_1\) and RMSE for Depth Upsampling methods over ARKitScenes.
Another topic is that the HR ground truth depth map in ARKitScenes is a projection of laser scans that were taken from different viewpoints, therefore occlusions may cause some parts of the image to lack depth information. Hence, some of the methods in the existing research need to be adapted to handle the special value of no-depth. Specifically, the Structural Similarity Index Measure (SSIM) loss used by MSPF \cite{44} cannot be used over ARKitScenes, as it is a full-reference method, requiring the information in all pixels without masking. In addition, the edge loss used by MSPF operates on the entire image and therefore needed to be changed to a more robust loss. We opted to use the edge loss suggested by \cite{45}. More details about these adaptations can be found in the supplementary material.

**Experimental Results.** We would like to compare the results of existing methods on ARKitScenes. For these experiments we reproduced three classical approaches for depth upsampling - naïve Bilinear interpolation, Joint Bilateral Upsampling (JBU) \cite{36} and Fast Guided Global Interpolation (FGI) \cite{37}, as well as the two mentioned modern DNN-based solutions - MSG \cite{35} and MSPF \cite{44}. In order to have cleaner data for training, we removed frames where regions of missing information took more than 40% of the HR depth map. Also, in order to avoid issues originating from the ground truth registration process, we ignore frames at which the Root Mean Square Error (RMSE) between the LR and a downscaled HR depth map is more than 20cm. After applying those 2 filters we remain with approximately 400K frames which are 65% of all the data.

With a target to improve run-time while maintaining high diversity between frames, we subsampled the dataset by taking a single frame every two seconds. This led us to train the models with 39k frames from the train split and evaluate on a different 6k frames from the validation split. The train and validation split were taken from completely different houses. More details on the experiments settings along with more examples can be found in the supplementary material. A visual comparison shown in Figure 5 along with many more examples in the supplementary material clearly show how the methods that were trained on ARKitScenes (MSG and MSPF) produce sharper edges and more realistic structure in the predicted image. It is also supported by comparing the absolute difference ($\ell_1$) and the Root Mean Square Error (RMSE) between the methods on Table 3 in which MSG and MSPF outperform the classical methods by 7 to 12% in RMSE, and 25 to 35% in $\ell_1$. While those results are encouraging, we believe that future research in this field could leverage this new dataset to suggest more sophisticated methods benefiting from the quantities and nature of this dataset to overcome the difficulties in upsampling a noisy LR image in real-world scenarios.

**5 Conclusions**

We presented ARKitScenes, it is not only the first dataset that is captured with Apple LiDAR sensor, but also to the best of our knowledge the largest indoor RGB-D dataset ever collected with a mobile device. We showed how our dataset can be used for two downstream computer vision tasks of 3D Object Detection and Color Guided Depth Upsampling. We believe ARKitScenes will enable the research community to push the boundaries of existing state of the art and develop technologies that better generalizes to realistic scenarios.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In Section 4.2 and the supplementary we discuss the limitation of occlusions in the ground truth and how to adapt existing algorithms to it
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main
       experimental results (either in the supplemental material or as a URL)? [Yes] Sample
       data shared via URL, full data will be shared upon acceptance
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they
       were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experi-
       ments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g.,
       type of GPUs, internal cluster, or cloud provider)? [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [Yes] Currently the license that we plan to
       apply to this dataset is a non-commercial creative commons license
   (c) Did you include any new assets either in the supplemental material or as a URL?
       [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data
       you’re using/curating? [Yes] Section 3
   (e) Did you discuss whether the data you are using/curating contains personally identi-
       fiable information or offensive content? [Yes] Section 3

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if
       applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review
       Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount
       spent on participant compensation? [N/A]