Abstract

This paper introduces a pipeline to parametrically sample and render static multi-task vision datasets from comprehensive 3D scans from the real-world. In addition to enabling interesting lines of research, we show the tooling and generated data suffice to train robust vision models. Familiar architectures trained on a generated starter dataset reached state-of-the-art performance on multiple common vision tasks and benchmarks, despite having seen no benchmark or non-pipeline data. The depth estimation network outperforms MiDaS and the surface normal estimation network is the first to achieve human-level performance for in-the-wild surface normal estimation—at least according to one metric on the OASIS benchmark.

The Dockerized pipeline with CLI, the (mostly python) code, PyTorch data loaders for the generated data, the generated starter dataset, download scripts and other utilities are all available through our project website.

1. Introduction

This paper introduces a pipeline to bridge the gap between comprehensive 3D scans and static vision datasets. Specifically, we implement and provide a platform that takes as input one of the following:

- a textured mesh,
- a mesh with images from an actual camera/sensor,
- a 3D pointcloud and aligned RGB images,

and generates a multi-task dataset with as many cameras and images as desired to densely cover the space. For each image, there are 21 different default mid-level cues, shown in Fig. 1. The software makes use of Blender [16], a powerful physics-based 3D rendering engine to create the labels, and exposes complete control over the sampling and generation process. With the proliferation of reasonably-priced 3D sensors (e.g. Kinect, Matterport, and the newest iPhone), we anticipate an increase in such 3D-annotated data.

In order to establish the soundness for training computer vision models, we used our pipeline to annotate several existing 3D scans and produce a medium-size starter dataset of mid-level cues. Samples of the data and different cues are shown in Fig. 5. Standard models trained on
this starter dataset achieve state-of-the-art performance for several standard computer vision tasks. For surface normal estimation, a standard UNet [45] model trained on this starter dataset yields human-level surface normal estimation performance on the in-the-wild dataset OASIS [12], even though the model never saw OASIS data during training. For depth estimation, our DPT-Hybrid [41] is comparable to or outperforms state-of-the-art models such as MiDaS DPT-Hybrid [42, 41]. The qualitative performance of these networks (shown in Figs. 6, 7) is often better than the numbers suggest, especially for fine-grained details.

We further provide an ecosystem of tools and documentation around this platform. Our project website contains links to a Docker containing the annotator and all necessary libraries, PyTorch [39] dataloaders to efficiently load the generated data, pretrained models, scripts to generate videos in addition to images, and other utilities.

We argue that these results should not be interpreted narrowly. The core idea of the platform is that the “sectors of the ambient [light-field] array are not to be confused with temporary samples of the array” (J. J. Gibson [21]). That is, static images only represent single samples of the entire 360-degree panoramic light-field environment surrounding an agent. How an agent or model samples and represents this environment will affect its performance on downstream tasks. The proposed platform in this paper is designed to reduce the technological barriers for research into the effect of data sampling practices and into the interrelationships between data distribution, data representation, models, and training algorithms. We discuss directions here and analyze a few illustrative examples in the final section of the paper.

First, the pipeline proposed in this paper provides a possible pathway to understand such sampling effects. That is, the rendering pipeline offers complete control over (heretofore) fixed design choices such as camera intrinsics, scene lighting, object-centeredness [40], the level of “photographer’s bias” [6], data domain, and so on. This makes it possible to run intervention studies (e.g. A/B tests), without collecting and validating a new dataset or relying on a post-hoc analysis. As a consequence, this provides an avenue for a computer vision “dataset design guide”.

Second, vision is about much more than semantic recognition, but our datasets are biased towards that as the core problem. The best-studied, most diverse and largest dataset (>10M images) generally contains some form of textural/class labels [18, 51] and only RGB images. On the other hand, datasets for most non-classification tasks remain tiny by modern standards. For example, the indoor scene dataset NYU [47], still used for training some state-of-the-art depth estimation models [62], contains only 795 training images—all taken with a single camera. The pipeline presents a way to generate datasets of comparable quality for non-recognition tasks.

Third, the generated data allows “matched-pair experimental design” that simplifies study into the interrelationships of different tasks, since the pipeline produces labels for every sample. In particular, it helps to avoid issues like the following: suppose a model trained for object classification on ImageNet transfers to COCO [32] better than a model trained for depth estimation on NYU [47]—is that due to the data domain, the training task, the diversity of camera intrinsics, or something else?

Existing matched-pair datasets usually focus on a single domain (indoor scenes [64, 47, 3, 50], driving [20, 17], block-worlds [24], etc.) and contain few cues [17, 47, 3, 50]. The provided starter dataset may be a better candidate for this research than these existing datasets, since it contains over 14.5 million images from different domains (more than the full ImageNet database), contains many different cues (e.g. for depth, surface normals, curvature, panoptic segmentation, and so on), and models trained on this dataset reach excellent performance for several tasks and existing benchmarks. We demonstrate the value of such matched-pairs data in Sec. 5.3.

Though our pipeline is designed to facilitate understanding the principles of dataset design, vision beyond recognition, the interrelationships between data, tasks, and models, this paper does not extensively pursue those questions themselves. It provides a few analyses, but these are merely intended as illustrative examples. Instead, the paper introduces tooling designed to facilitate such research as 3D data becomes more widely available and the capture technology improves. On our website, we provide a documented, opensourced, and Dockerized annotator pipeline with a convenient CLI, runnable examples, the starter dataset, pretrained models, PyTorch dataloaders, and code for the paper (including annotator and models).

2. Related Work

In this section we examine related datasets and other approaches. A thorough review would take more space than we have, so we restrict our attention to only the most relevant groupings.

Static 3D Datasets. The past few years have witnessed an uptick in the number of mesh-based datasets, thanks largely to the availability of reasonably-priced 3D scanners. Each dataset in the current crop, though, usually consists of scenes in a restricted domain. Prominent examples of indoor building datasets include Stanford Building Dataset (S3DIS) [5], Matterport3D [9], Taskonomy [64], Replica [50], 2D-3D-Semantic [4], Habitat-Matterport [36], and Hypersim [44]. Other datasets contain primarily outdoor scenes, usually driving—such as CARLA [20], GTA5 [43]—or other narrow domains such as the aptly-named Tanks and Temples [28] dataset. Models trained on such scene-level views often do not generalize
to object-centric views (see Fig. 7), but existing datasets with high-resolution object meshes do not include 2D images samples [1, 8].

Other recent datasets aim to link diverse monocular 2D images and corresponding 3D meshes, but take the reverse approach of this paper by using hand-annotation to create meshes from single-view in-the-wild RGB samples [12, 13]. This labeling process is expensive and time-consuming, and crucially does not allow regenerating the image dataset. In Sec. 4.3, we consider our pipeline vs. OASIS, one of the largest and most diverse of these benchmarks, and demonstrate that models trained on our starter dataset already match human-level performance on OASIS—outperforming the same architectures models trained on OASIS itself.

Vision-Focused Simulators. Like our platform, simulators typically take a textured mesh as the representation of the scene and aim to produce realistic sensory inputs [36, 59]. While spiritually similar to the pipeline proposed in this paper, the current generation of simulators is designed first and foremost to train embodied agents. They prioritize rendering speed and real-time mechanics at the cost of photorealism and cue diversity [25, 38]. Extending such simulators to handle additional cues or to parametrically render out vision datasets often requires writing new components of the simulator codebase (usually in C++, CUDA, or OpenGL), a surmountable but unpleasant barrier to entry. In contrast, our platform extends Blender which “supports the entirety of the 3D pipeline” [15] and provides Python bindings that will be intuitive to most vision researchers, and we implement many of these cues and sampling methods out-of-the-box. In short, we provide a bridge between simulators and static vision datasets.

Multi-Task Datasets. Vision-based multi-task learning (MTL), like computer vision in general, shows a general bias towards recognition. MTL datasets often take different shades of classification as the core problem of interest [30, 57, 35]. In particular, MTL literature often focuses on binary attribute classification in specialized domains, such as Caltech-UCSD Birds [58] or CelebA [34]. Visual MTL datasets that contain non-recognition tasks often contain only a single domain or a few tasks (NYU [47], CityScapes [17] or Taskonomy [64]). Sometimes, MTL papers take mix datasets for a “single” task and consider each dataset as a different task [33, 42, 31, 41].

In general, the multi-task learning literature has not converged on a standardized definition of the setting or dataset. Recent work has demonstrated that MTL methods developed on existing datasets seem to specialize to their respective development set and do not perform well on large, realistic datasets, or on other tasks [55, 56, 65]. This underscores the importance of developing realistic training setting and datasets that generalize to real-world scenarios.

Data Augmentation + Domain Randomization. Data augmentation is a way to modify the data or training regimen so that the trained model exhibits desirable invariances (or equivariances). Specifically, any transformation of sensor inputs that determines a corresponding (possibly identity) transformation on the label can be used as “augmented” data. Simple 2D augmentations such as 2D affine transformations, crops, and color changes are by far the most common in computer vision [10, 22], since they can be used even when datasets lack 3D geometry information. In robotics and reinforcement learning where 3D simulators are more standard, data augmentation was introduced as “domain randomization” [53], and common augmentations include texture and background randomization on the scene mesh. Recently, [19] introduced a Blender-based approach for doing domain randomization and creating static datasets of RGB, depth, and surface normals from SunCG [52].

Our pipeline makes all these augmentations available for static computer vision datasets: not just flips/crops/texture randomization, but also dense viewpoints, multi-view consistency, Euclidean transforms, lens flare, etc.). We implement and examine depth-of-field augmentation in Sec. 5.1.
3. Pipeline Overview

We call our pipeline Omnidata as it aspires to encapsulate comprehensive scene information (“omni”) in the generated “data”. Try a live example here to get acquainted with the pipeline. The example uses the CLI and a YAML-like config file to generate images from a textured mesh in Replica [50].

**Inputs:** The annotator operates upon the following inputs:
- Untextured Mesh (.obj or .ply)
- Either: Mesh Texture or Aligned RGB Images
- Optional: Pre-Generated Camera Pose File

A 3D pointcloud can be used as well: simply mesh the pointcloud using a standard mesher like COLMAP [46]. An example of meshing and using a 3D pointcloud with the annotator, as well as a more complete description of inputs are available in the supplementary.

**Outputs:** The pipeline generates 21 mid-level cues in the initial release. All labels are available for all generated images (or videos). Fig. 1 provides a visual summary of the different types of outputs. A detailed description of the default mid-level cues and additional outputs provided by the Omnidata annotator is included in the supplementary.

3.1. Sampling and Generation

In this section we provide a high-level outline of the generation and rendering process (see Fig. 2), deferring full details to the supplementary.

**First**, the annotator generates camera locations (Fig. 2 II) and points-of-interest (Fig. 2 III) along the mesh.

**Second**, for each camera and each point-of-interest, it creates a view from that camera fixated on the point (three fixated views are depicted in the lower part of Fig. 3).

**Third**, for each space-point-view triplet, the annotator renders (Fig. 4) all the mid-level cues (Fig. 2 IV).

Each step is elaborated next.

![Figure 4: DAG of processing pipeline. The pipeline uses some of the mid-level cues to produce others. The ordering of this process (for image-like cues) is shown by the DAG.](image)

**Camera and Point Sampling:** Camera locations can be provided (if the mesh comes with aligned RGB), or as in Fig. 2 II, the annotator generates cameras in each space so that cameras are not inside or overlapping with the mesh (default: cameras generated via Poisson-disc sampling to cover the space). Points-of-interest are then sampled from the mesh with a user-specified sampling strategy (default: uniform sampling of each mesh face and then uniform sampling on that face). Cameras and points are then filtered so that each camera sees at least one point and each point is seen by at least some user-specified minimum number of cameras (default: 3).

**View Sampling:** The annotator provides two default methods for generating views of each point. The first method (wide-baseline) generates images while the second, (smooth-trajectory mode) generates videos.

- **Wide-baseline multi-view:** A view is saved for each space-camera-point combination where there is an unobstructed line-of-sight between the camera center and the point-of-interest. The camera is fixated on the point-of-interest, as shown in Fig. 3, bottom.

- **Smooth trajectory sampling:** For each point of interest, a subset of cameras with a fixated view of the point are selected, and a smooth cubic-spline trajectory is interpolated between the cameras. Views of the point are generated for cameras at regular intervals along this trajectory (see Fig. 3, top).

**Rendering mid-level cues:** Since no single piece of software was able to provide all mid-level cues, we created an interconnected pipeline connecting several different pieces of software that are all freely available and open-source. We tried to primarily use Blender (a 3D creation suite), since it has an active user and maintenance community, excellent documentation, and python bindings for almost everything. Used by professional animators and artists, it is generally well-optimized. The overall pipeline is fairly complex, so
we defer a full description to the supplementary. The order of cue generation is shown in Fig. 4. The full code is available on our website.

Performance: The annotator generates labels in any resolution. Each space+point+view+cue label in the starter dataset (512 × 512) typically takes 1-4 seconds on server or desktop CPUs and can be parallelized over many machines.

3.2. Ecosystem Tools

To simplify adoption, the following tools are available on our website and the associated GitHub repository:

Pipeline code and documentation.
Docker containing the annotator and properly linked software (Blender [15], compatible Python versions, MeshLab [14], etc.).
Dataloaders in PyTorch for correctly and efficiently loading the resulting dataset
Starter dataset of 14.5 million images with associated labels for each task

Convenience utilities for downloading and manipulating data and automatically filtering misaligned meshes (description and sensitivity analysis in the supplementary).
Pretrained models and code, including the first publicly available implementation of MiDaS [42] training code.

4. Starter Dataset Overview

We provide a relatively large starter dataset of data annotated with the Omnidata annotator. The dataset comprises roughly 14.5 million images from scenes that are both scene- and object-centric. Fig. 5 shows sample images from the starter dataset along with 12 of the 21 mid-level cues provided. Cues that are not present in the original dataset are indicated with a red border. We evaluate this starter dataset on existing benchmarks in Sec. 4.3. Note that the dataset could be straightforwardly extended to other existing outdoor and driving datasets such as GTA5 [43], CARLA [20], or Tanks and Temples [29].

4.1. Datasets Included

The starter data was created from 7 mesh-based datasets:

Indoor scene datasets: Replica [50], HyperSim [44], Taskonomy [64], Habitat-Matterport (HM3D)
Aerial/outdoor datasets: BlendedMVG [61]
Diagnostic/Structured datasets: CLEVR [24]

Object-centric datasets: To provide object-centric views in addition to scene-centric ones, we create a dataset of Google Scanned Objects [1] scattered around buildings from the Replica [50] dataset (similar to how ObjectNet [7] diversified images for image classification). We used the Habitat [36] environment to generate physically plausible scenes, and generated different densities of objects. Examples of images are shown in Fig. 5, and a full description of the generation process is available in the supplementary.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>Val</th>
<th>Test</th>
<th>Spaces</th>
<th>Train</th>
<th>Val</th>
<th>Test</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEVR</td>
<td>60,000</td>
<td>6,000</td>
<td>6,000</td>
<td>1 0 0 72,000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replica</td>
<td>56,783</td>
<td>23,725</td>
<td>23,889</td>
<td>10 4 4 45,150</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Replica + GSO</td>
<td>107,404</td>
<td>43,450</td>
<td>42,665</td>
<td>10 4 4 31,167</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HyperSim</td>
<td>59,543</td>
<td>7,386</td>
<td>7,690</td>
<td>365 46 46 74,619</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Taskonomy</td>
<td>3,416,314</td>
<td>538,567</td>
<td>629,581</td>
<td>379 75 79 684,052</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BlendedMVG</td>
<td>79,023</td>
<td>16,787</td>
<td>16,766</td>
<td>341 74 73 112,576</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Habitat-Matterport</td>
<td>8,470,855</td>
<td>1,061,021</td>
<td>-</td>
<td>800 100 - 564,328</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Component dataset statistics. Breakdown of train/val/test split sizes in each of the components of the starter dataset.

4.2. Dataset Statistics

The starter dataset contains 14,601,449 images from 2,414 spaces. Views are both scene- and object-centric, and they are labeled with each modality listed in Fig. 1. Camera field-of-view is sampled from a truncated normal distribution between 30° and 125° with mean 77.5°, and camera
roll is uniform in $[-10^\circ, 10^\circ]$. Tab. 1 contains data broken down to sub-datasets.

### 4.3. Soundness for Existing Computer Vision

We demonstrate that the generated dataset is capable of training standard, modern vision systems to state-of-the-art performance on existing benchmarks. Once we have established that the models can be trusted, we further provide a few transfer experiments to quantify how related the different component datasets are.

We show that the models trained on a 5-dataset portion of the starter dataset (4M images) for depth and surface normal estimation have state-of-the-art performance on the in-the-wild OASIS benchmark. To demonstrate the effectiveness of the pipeline for semantic tasks, we show that the predictions from a network trained for panoptic segmentation on a smaller 3-dataset portion (1M images) are of similar quality to models trained on COCO [32]. Full experimental details and more results are available in the supplementary.

Table 2: Zero-shot depth estimation. On NYU and OASIS, a DPT-Hybrid trained on the Omnidata starter dataset is comparable or better than the same model trained on existing depth datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Data</th>
<th>L1 Error (%)</th>
<th>$\delta &gt; 1$ (%)</th>
<th>$\delta &gt; 1.25$ (%)</th>
<th>$\delta &gt; 1.25^2$ (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiDaS v3 [41]</td>
<td>Omnidata</td>
<td>0.3957</td>
<td>22.03</td>
<td>67.28</td>
<td>52.38</td>
</tr>
<tr>
<td>Omnidata</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Zero-shot surface normal estimation on OASIS. A UNet trained on the Omnidata starter dataset matched or outperformed models trained on OASIS itself, and it matched human-level $AUC_p$.

Table: Zero-shot depth estimation. On NYU and OASIS, a DPT-Hybrid trained on the Omnidata starter dataset is comparable or better than the same model trained on existing depth datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Data</th>
<th>Mean</th>
<th>Median</th>
<th>% Within $\delta$</th>
<th>Relative Normal $AUC_p$</th>
<th>$AUC_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MiDaS v3 [41]</td>
<td>NYU [47]</td>
<td>35.32</td>
<td>29.21</td>
<td>14.23</td>
<td>57.72</td>
<td>51.31</td>
</tr>
<tr>
<td>PBRS [66]</td>
<td>NYU [47]</td>
<td>35.32</td>
<td>32.14</td>
<td>11.59</td>
<td>52.14</td>
<td>50.46</td>
</tr>
<tr>
<td>UNet [45]</td>
<td>SunCG [29]</td>
<td>35.42</td>
<td>28.70</td>
<td>12.31</td>
<td>58.51</td>
<td>52.15</td>
</tr>
<tr>
<td>Human (Approx.)</td>
<td></td>
<td>35.42</td>
<td>28.70</td>
<td>12.31</td>
<td>58.51</td>
<td>52.15</td>
</tr>
</tbody>
</table>

Monocular depth estimation: The current best approach for depth estimation is to aggregate multiple smaller datasets and train with scale- and shift-invariant losses [42, 41] to handle the different unknown depth ranges and scales. As of this writing, the DPT-based [41] models from “MiDaS v3.0” [41] define the state-of-the-art, especially on NYU [47]. We adopt a similar setting to MiDaS v3.0, but train on a 5-dataset portion of our starter dataset instead of their 10-dataset mix.

As in [41], we evaluate zero-shot cross-dataset transfer with test predictions and GT aligned in scale and shift in inverse-depth space. Tab. 2 shows that the DPT-Hybrid trained on our starter dataset outperforms MiDaS DPT-Hybrid on both the test set of NYU [47] and the validation split of OASIS (the test GT is not available). The error metrics use $\delta = max(d, d^*)$ where $d$ and $d^*$ are aligned depth and ground truth. Our model better recovers the fine-grained details and true shape of the objects—this is especially clear in the surface normals extracted from the predictions (last 2 rows of Fig. 6). Full details, code, and more qualitative results are available on our website.

Surface normal estimation: Similar to the existing models on the surface normal task of OASIS, we train a vanilla UNet [45] architecture (6 down/6 up, similar to [63]) with angular & L1 losses, light 2D data augmentation, and input resolutions between 256 and 512. We use Adam [26] with LR $10^{-4}$ & weight decay $2 \times 10^{-6}$. The results in Tab. 3 indicate that our model matched human-level performance on OASIS $AUC_p$. On most of the remaining metrics, it outperformed related models trained on other datasets (including OASIS itself) and models with architectures specifically designed for normals estimation (PBRS). Fig. 7 shows that our model qualitatively performs much better on selected images than is indicated by the numbers, which may be because the standard metrics do not align with perceptual quality as “uninteresting” areas (walls, floors) dominate the

---

1MiDaS v3.0 also uses MTAN [33] for dataset balancing, and though in Sec. 5.3 we examine MTAN (it indeed helped on our dataset), we used here a naive sampling strategy in order to be consistent with the majority of the other models in this paper.
Figure 7: Qualitative results of zero-shot surface normal estimation. The 3 models are trained on OASIS[12], Full Taskonomy[63, 64], and our starter set. Queries are from 3 different datasets (OASIS, Taskonomy, GSO+Replica) in addition to some external queries in the last 2 columns (no ground truth available) which show the generalization of the models to external data [best viewed zoomed in].

Figure 8: Qualitative results of panoptic segmentation with PanopticFPNs [27] trained on COCO [32] and Omnidata. The Omnidata model trained jointly on Taskonomy, Replica, and Hypersim shows good out-of-distribution performance on indoor scenes without people.

4.3.1 Dataset Relatedness
To estimate how the components of the starter dataset are related, we use zero-shot cross-dataset transfer performance for surface normal and panoptic segmentation models trained on different components. Tab. 4 shows that each single model performs well on its corresponding test set, but typically generalizes poorly. The models trained on larger splits perform better overall (see supplementary). The model trained on the largest set achieved the best average performance (harmonic mean 25.8% and 30.3% better than best single-dataset models for surface normal estimation and panoptic segmentation). The ranking of transfers depended on the task, which might be due to the sparse panoptic labels on Taskonomy (from the followup paper [2]), but we believe the dependency is true in general.

Table 4: Inter-dataset domain transfer performance for surface normal estimation and panoptic segmentation. Models trained on each individual dataset and Omnidata are evaluated on test splits of the starter set. The harmonic mean across datasets is shown in the last column. (* PQ on things classes only, as Taskonomy does not feature stuff labels.)

5. Illustrative Data-Focused Analyses
Now that we have established that the annotator produces datasets capable of training reliable models, what analyses can we do with such datasets? We survey a few examples here, but they are not intended to be comprehensive (Sec. 1).

5.1. New 3D Data Augmentations
Data augmentation is used to address shortcomings in model performance and robustness. For example, models trained only on images captured with narrow apertures (e.g. NYU or Taskonomy) tend to perform poorly on images taken with a wide aperture (i.e. strong depth-of-field), and augmenting with 2D Gaussian blur is used to improve model performance on unfocused portions of images. The approach is common enough that 2D blur was included in the Common Corruptions benchmark [23]. Because the full scene geometry is available for our starter dataset, it is possible to do the data augmentation in 3D (image refocusing) instead of 2D (flat blurring). Fig. 9 shows an example of what 3D “image refocusing” augmentation on our dataset looks like. In the supplementary, we show that models
trained for surface normal estimation using only 3D augmentation were more robust to both 2D blurring and 3D refocusing than those trained with 2D augmentation.

![Query Image](image1) Shallow Focus Mid Focus Far Focus

Figure 9: Image refocusing augmentation on Taskonomy. Portions of the image that are in focus are highlighted in red [best viewed zoomed in].

5.2. Mid-Level Cues as Inputs: Are They Useful?

Is there an advantage in using multiple sensors or non-RGB representations of the environment? Instead of predicting mid-level cues as the downstream task (i.e. multi-task learning), multiple cues could also be used as inputs (if relevant sensors are available) or specified as intermediate representation (with the labels being used as supervision only during training, i.e. PADNet [60]).

Tab. 5 demonstrates that using these additional cues in the latter 2 ways can improve performance on the original test set and also on unseen data. In this experiment, we train HRNet-18 [52] backbones for semantic segmentation using a single component dataset (10 spaces from Replica) and evaluate them on Replica, Hypersim and Taskonomy (tiny split). Relative to using only RGB inputs and the semantic segmentation labels, cross-entropy performance improves across the board when treating the cues as sensors (23%, 34% and 30%) or using them as intermediate representations (13%, 17%, and 19%). Adding more cues seems to help. Full experimental settings in the supplementary.

Future work could further analyze how the effectiveness of these different methods change with dataset size, which cues to use, how many additional images a mid-level cue is worth, and on the relative importance of getting more data from the same scene vs. adding data from new scenes.

<table>
<thead>
<tr>
<th>Method</th>
<th>Semantic Segmentation</th>
<th>3D Keypoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single task</td>
<td>(All Above) + Normals</td>
<td>Ours NYU [47] CityScapes [17], Taskonomy [54]</td>
</tr>
<tr>
<td>MTL baseline</td>
<td>(All Above) + 2D Edges</td>
<td>80.69 4</td>
</tr>
<tr>
<td>MTAN [33]</td>
<td>(All Above) + 2D Edges</td>
<td>83.00 2</td>
</tr>
<tr>
<td>Cross-stitch [37]</td>
<td>(All Above) + Normals</td>
<td>85.12 1</td>
</tr>
</tbody>
</table>

Table 6: Multi-task training methods do not show a clear ordering. Within-task, rankings between different methods were indistinguishable from random orderings (i.e. ρ = 0). Between tasks, rankings on Sem. Seg. were anti-correlated with rank on the 3D Keypts (ρ = -0.4). Both conclusions were strengthened after controlling for training setups.

6. Conclusion

This paper introduces a pipeline, Omnidata, to bridge the gap between real-world 3D scans and static vision datasets. The pipeline makes it possible to generate large and diverse datasets for many tasks with greater control over the data generation parameters, and to pursue fundamental lines of research such as how the data sampling strategy and collected cues impact model reliability. We examine a few example analyses, on a starter dataset generated from the pipeline. To demonstrate the utility and quality of the data, we trained a few standard vision models on this starter data and they achieved state-of-the-art performance on multiple computer vision tasks.

Acknowledgement: We thank Roman Bachmann for the discussions, implementations, and experimental help.
References

[1] Ignition app.


[55] Simon Vandenhende, Stamatios Georgoulis, Bert De Brabandere, and Luc Van Gool. Branched multi-task net-


