ODIN: A Single Model for 2D and 3D Perception

Thank you for your feedback. The reviewers find the idea of segmenting both 2D images and 3D point clouds with a single architecture interesting (PhCo, EST1, vCUv), with "impressive" performance on several 3D datasets (vCUv, EST1) and in an embodied setup (vCUv).

[PhCo] ***"ODIN lags behind MAFT [28] by 10 points in ScanNet."*** MAFT operates over 3D point clouds sampled from ScanNet meshes, similar to Mask3D. As explained in L419-425, L126-L143, this provides an advantage over methods that use sensor point clouds, like ODIN, **because labels are annotated over mesh 3D points, and are misaligned to the sensor 3D point cloud (and corresponding RGB 2D point features), as also pointed out in previous works [21, 35]** (EST1). MAFT is concurrent to ODIN and its contribution is on devising different methods for object query initialization and refinement, which ODIN can incorporate. Thus, MAFT is complementary to our approach (L382-387), which targets to forego benchmark 3D meshes and utilize pre-trained 2D backbones for 3D segmentation.

[vCUv] ***"Perception is more than segmentation."*** Fair point. We will replace "perception" with *"segmentation"* in our paper title and all other instances in the paper.

[vCUv] ***"ODIN running "directly on sensor data" is misrepresentative of the need for pose."*** We agree. We will clarify that ODIN needs additional post-processing for obtaining poses and delete lines about running directly on sensor data. As you point out, *the main difference between the methods is how much post-processing is needed, not whether it is needed or not.* For example, ScanNet utilizes BundleFusion exclusively for obtaining poses, not meshes, which are obtained by VoxelFusion, followed by various post-processing steps. Additionally, many real-world applications use techniques like Visual Odometry, SLAM, ICP, SfM, IMU, and GPS sensors, none of which mandate 3D mesh reconstruction.

[vCUv] ***"Jointly 2D-3D training drops 2D performance. Data balancing?"*** Currently, training jointly over 2D and 3D helps performance in 3D without dramatically dropping performance in 2D. Using COCO:ScanNet ratio of 2:1 yields 46.1 mAP on ScanNet and 42.5 mAP on COCO compared to 1:1 ratio in the paper which yielded 48.3 mAP on ScanNet and 40.7 mAP on COCO. We will add the following to our limitations: *"Our results suggest a competition between 2D and 3D segmentation performance when training ODIN jointly on both modalities. Exploring ways to make 2D and 3D training more synergistic is a promising avenue for future work."*

[EST1] ***" In Figure 2, do RGB-D and single RGB images share input layer, backbone and mask decoder head?"*** There's no dedicated "input layer", this only signifies the entry point of the inference process. From the backbone, the 2D ResBlocks are shared across both RGB and RGB-D images, while the 3D RelPos Attn blocks are only used by the RGB-D images. Mask decoder head weights are fully shared across both RGB and RGB-D images, and differentiation between RGB and RGB-D is based solely on the positional encodings. [EST1] ***"How do you combine 2D and 3D features?"*** RGB-D and RGB images are fed separately in different forward passes *but they share the neural architecture weights*, as explained above.

[EST1] ***"Table 4: Why and how joint 2D-3D training helps?"*** Joint 2D-3D training helps through weight sharing across both 2D and 3D segmentation tasks, which helps fight overfitting on the smaller-scale ScanNet dataset. ScanNet dataset is much smaller than COCO and thus 2D-3D co-training does not improve performance in the 2D COCO benchmark.

[vCUv] ***"Inference time on ODIN vs Mask3D"*** We follow Mask3D and only include model inference time without the data-loading time in our reported inference time. We use batch size=1 for both ODIN and Mask3D. All the N views are processed in parallel resulting in a single forward pass through the model for ODIN. The 2D-3D and 3D-2D projection operations involve cheap computations like reshaping or matrix multiplication. As discussed in Section 1.4 of the appendix, Mask3D with sensor point cloud is slower than with mesh point cloud because at the same voxel size (0.02m), more voxels are occupied in sensor point cloud (110k on avg.) compared to mesh point clouds (64k on avg.) as mesh-cleaning sometimes discards large portion of the scene. The transfer of features from the sensor point cloud to the mesh point cloud adds an extra 7 ms.

[vCUv] ***"Is feature transfer from sensor to mesh just a different "Upsample layer""*** Yes, a different upsampling layer, where the upsampling targets come from mesh point cloud instead of sensor point cloud.

[vCUv] ***"Performance of ODIN on mesh-based depth? Mask3D is not tuned for AI2THOR"*** ODIN with mesh depth obtains 48.3% mAP on ScanNet, compared to 45.7% mAP with sensor depth. Note that this mesh-rendered depth still has misalignments with the sensor RGB image, where our point features come from. Rendered RGB images have significant rendered artifacts which make them unsuitable for pre-trained backbones. Additionally, depth rendered from ScanNet mesh typically has large holes due to missing scene regions which got dropped during mesh-cleaning. For Mask3D on AI2THOR, we tried several training schedules and hyperparameters and kept the best results, while training it for more than a week to ensure convergence. Besides, ODIN outperforms Mask3D on same sensor depth on ScanNet and both sensor and mesh depths on ScanNet200.