OmniNOCS: A unified NOCS dataset and model for 3D lifting of 2D objects

Akshay Krishnan[1,2], Abhijit Kundu[1], Kevis-Kokitsi Maninis[1], James Hays[2], and Matthew Brown[1]

[1] Google Research[†]
[2] Georgia Institute of Technology

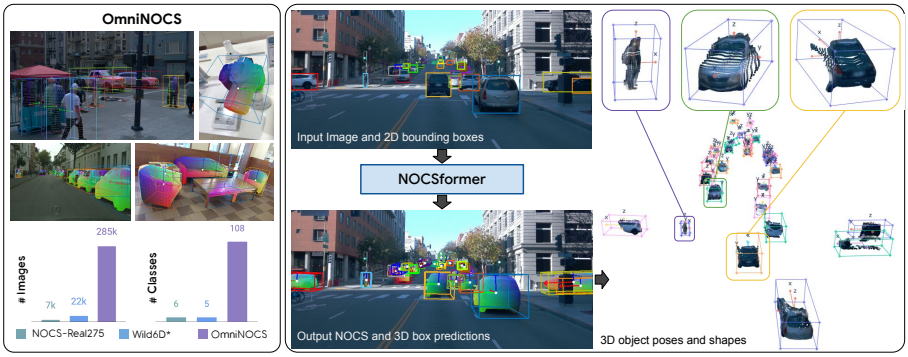

Fig. 1: We introduce **OmniNOCS**, a large-scale dataset with Normalized Object Coordinates (NOCS), instance masks, and 3D box annotations for objects across several classes, domains, and cameras. We also propose **NOCSformer**, a model trained on OmniNOCS that lifts each 2D object bounding box in an image to its corresponding 3D oriented box (pose) and 3D pointcloud (shape).

Abstract. We propose OmniNOCS, a large-scale monocular dataset with 3D Normalized Object Coordinate Space (NOCS) maps, object masks, and 3D bounding box annotations for indoor and outdoor scenes. OmniNOCS has 20 times more object classes and 200 times more instances than existing NOCS datasets (NOCS-Real275, Wild6D). We use OmniNOCS to train a novel, transformer-based monocular NOCS prediction model (NOCSformer) that can predict accurate NOCS, instance masks and poses from 2D object detections across diverse classes. It is the first NOCS model that can generalize to a broad range of classes when prompted with 2D boxes. We evaluate our model on the task of 3D oriented bounding box prediction, where it achieves comparable results to state-of-the-art 3D detection methods such as Cube R-CNN. Unlike other 3D detection methods, our model also provides detailed and accurate 3D object shape and segmentation. We propose a novel benchmark for the task of NOCS prediction based on OmniNOCS, which we hope will serve as a useful baseline for future work in this area. Our dataset and code will be at the project website: https://omninocs.github.io.

1 Introduction

Predicting the 6 Degree-of-Freedom (6DoF) pose and shape of objects from images is a crucial problem in 3D scene understanding. Robots need to understand the location, shape, and orientation of various objects to grasp and interact with them. Self-driving

[†]Now at Google DeepMind

vehicles need to understand the location and heading of vehicles, pedestrians, and other objects on the road. In particular, the 3D orientation of these objects is crucial to predict their future behavior. It is also important in AR/VR applications, as it allows users to interact with objects in meaningful ways. Predicting object shape and pose from monocular images is also a prerequisite to initialize panoptic 3D neural scene graph representations [26, 39] and for methods that track objects in videos [42]. Most of these applications require approaches that generalize across a wide range of object classes, environments, and camera models.

The problem of localizing 3D objects has been extensively studied through the lens of monocular 3D object detection or 6DoF pose estimation. The most commonly used approach is to represent objects as 3D cuboids, and train a model to regress the cuboid parameters from a 2D RoI of the object [7, 8, 27, 41, 51, 54]. An alternative method involves predicting corresponding 3D points in an object coordinate space followed by pose estimation using 3D-2D alignment [11, 12, 20, 33, 49]. However, all these methods are limited to narrow datasets collected on a single camera and context with typically very few object classes. Existing models are also trained separately for every dataset, preventing them from being used more widely.

Recent work [7] takes a step towards scalability by introducing the Omni3D dataset, which aggregates 3D detection datasets from different domains, and trains a Cube R-CNN model to regress 3D bounding boxes for 50 classes. However, Omni3D notably lacks consistently oriented, object-centric boxes, as the ground truth orientations are not canonical for object classes. Cube R-CNN [7] instead uses a Chamfer distance loss which is invariant to the predicted 3D cuboid orientation. This causes it to predict inconsistent object orientations, for example, often flipping the orientations of cars by 180°. Further, Omni3D is a detection dataset that does not provide object shape information.

Our work aims to overcome the above shortcomings, by providing a new large-scale dataset with consistent object-centric ground truth boxes along with detailed shape. We argue for the use of Normalized Object Coordinate Space (NOCS) as proposed in [49] as a 3D object shape representation. NOCS represents both *the canonical orientation and the shape* of the visible surface of the object, properties that are essential for real-world applications such as self-driving and robotics. It can also be used to estimate the 3D bounding box of the object. However, all existing work on NOCS [24,31,48,49] only train small models on small datasets with fewer than 10 classes. We address the lack of diverse NOCS datasets by proposing a large-scale monocular NOCS dataset, **OmniNOCS** that has NOCS annotations, instance segmentation, and canonically oriented bounding boxes for 97 object classes, containing 380k images from 10 different data sources, making it the largest and most diverse NOCS dataset currently available. OmniNOCS includes all of the data from Omni3D (KITTI [19], nuScenes [10], ARKitScenes [5], Objectron [1], Hypersim [44], SUN-RGBD [45]), with the addition of Cityscapes [15], virtual KITTI [9], NOCS-Real275 [49], and the Waymo Open Dataset [46].

Our work also introduces a new model, which we term "NOCSformer", that predicts NOCS coordinates and oriented 3D bounding boxes from monocular images and 2D detections for all the classes in OmniNOCS. NOCSformer leverages large self-supervised pretrained ViTs [17, 40]. It does not use any class-specific heads or parameters. This enables it scale to large vocabularies, and share information across semantically similar

object classes. In contrast, existing NOCS models use small NOCS heads with class-specific parameters that significantly limit their performance on large datasets like OmniNOCS. Apart from the NOCS predictions, NOCSformer also predicts the 3D size of the object, and 3D orientation using a learned PnP head. This allows NOCSformer to predict 3D oriented bounding boxes and object point cloud in metric scale for diverse object categories. Our experiments evaluate the quality of NOCS and bounding boxes predicted by NOCSformer in comparison to existing NOCS prediction or 3D detection models. We find that training on OmniNOCS allows NOCSformer to generalize to unseen datasets, even outperforming baselines trained on the target dataset in NOCS prediction accuracy.

In summary, our contributions are:

- **The OmniNOCS dataset**: A new dataset containing Normalised Object Coordinates for 380k images in 97 categories, an order of magnitude larger on both counts than existing NOCS datasets.
- **NOCSformer**: A novel transformer-based architecture for predicting object NOCS, mask, and size from input 2D boxes, utilizing pre-trained self-supervised ViT backbones. NOCSformer is the first NOCS model to generalize to vocabularies with 90+ classes and to unseen datasets, including images from internet collections.
- **OmniNOCS benchmark**: an evaluation framework with metrics for directly comparing different NOCS prediction algorithms on OmniNOCS, with baselines established via NOCSFormer.

2 Related work

The task of predicting 3D object pose has been studied both in the context of camera/object pose estimation (6DoF) and 3D bounding box estimation (6DoF pose + 3DoF size). A further distinction can be made between methods that regress directly to object pose / bounding box parameters, and methods that compute per pixel coordinates or depth as an initial step. We review each of these paradigms in the sections below.

2.1 3D localization by regressing bounding boxes

Direct approaches to 3D object detection and pose estimation involve networks that output rotation, translation and scale parameters directly. Several works have explored different parameterizations in this setting, e.g., PoseCNN [54] uses regression of translation via centre direction + distance maps (which enable detection even under occlusion), and quaternion representation of rotation. BB8 [41] uses segmentation followed by a CNN to regress to 2D projections of the 8 $\times$ 3D bounding box corners. Multi-view monocular approaches have also been proposed, e.g., DETR3D [52] which uses DETR-style attention to reason about object interactions. Several techniques also make use of existing 2D bounding box predictors, either as an input to a 3D lifting approach [34], or as a constraint on 3D box predictions [38].

Another group of works focuses on predicting the alignment of 3D CAD models within various modalities: images [21, 28], videos [30, 35], or 3D scans [2, 3]. These approaches typically determine the object's 3D translation, rotation, and size (9DoF) and find a CAD model with a similar visual appearance.

2.2 3D localization from model-to-image alignment

An alternative approach to 3D localization is to first predict correspondence, either between views or to a normalized space, and then reason over redundant correspondences to establish pose. This has been done with both sparse and dense correspondences, e.g., AutoShape [33] makes use of sparse 2D to 3D correspondences, with a learned shape model and sparse 2D keypoints. [20] uses pairwise semantic correspondence from ViT, to find pose between a reference image and a sequence of targets.

Several approaches use dense correspondences and the idea of Normalised Object Coordinate Spaces, or NOCS. In the original NOCS work, Wang et al. [49] use RGB-D data and 3D-3D correspondences for pose estimation. Follow-up work [11] estimates pose using PnP variants on 2D-3D correspondences. While NOCS [49] used the average spatial dimensions of the object category to define the object frame, [53] showed that an instance-specific NOCS coordinate frame can be used alongside a predicted instance size for the same purpose. Other methods combine coordinate regression and direct approaches, e.g., [48], which uses direct regression based on dense correspondences.

Similar to our work, MonoRUn [11] lifts 2D detections to 3D object coordinates without explicit class supervision. They also use a novel self-supervised coordinate regression training loss which obviates the need for detailed 3D ground truth. However, their representation does not include masks and therefore does not provide explicit object shape information. Their evaluation is also limited to 3 classes (car, pedestrian, cyclist) on the KITTI-Object test set. M3D-RPN [8] also has a single 3D head for multiple classes, jointly generating 2D and 3D bounding box predictions, though without shape information, and similarly limited to car, pedestrian and cyclist classes on KITTI.

2.3 Monocular object localization / pose estimation datasets

For 3D object detection from monocular RGB images, most existing works use a small number of classes on a single dataset. Cube R-CNN/Omni3D [7] contribute towards creating a general purpose monocular 3D object detector. Cube R-CNN performs well over six 3D datasets: KITTI, SUN RGB-D, ARKitScenes, Objectron, nuScenes and Hypersim. However, the method is class specific, with Cube R-CNN trained only on 50 classes, and unable to work in the open class setting.

Cube R-CNN also uses a Chamfer loss on the predicted 3D box corners to deal with the inconsistent coordinate frame annotations in the underlying Omni3D datasets. The lack of direct orientation supervision results in inconsistent orientation predictions (for example, the positive x axis may point to the front or the back of the car for different instances). This orientation inconsistency is problematic for methods that seek to build detailed 3D models of object categories as a downstream task [26, 39].

Other works are not limited to 3D boxes, and directly provide object shapes. These shapes come in the form of annotated point clouds (e.g., ScanNet [16]), or aligned CAD models (e.g., Scan2CAD [2], Pix3D [47], CAD-Estate [36]). Object point clouds are created by labeling points on scanned 3D scenes, a process that doesn't scale well to large datasets. Semi-automatically aligning CAD models is scale-able to a certain extent, but since it relies on retrieving existing models, the resulting shapes are rarely accurate, and the alignments are sensitive to deformations and movable parts. In contrast to these works, we propose a large-scale dataset of many different categories.

Dataset	NOCS GT	Real	#Images	#Classes	#Instances
CAMERA25 [49]	✓	✗	300k	6	184
NOCS-Real275 [49]	✓	✓	7k	6	24
Wild6D [18]	✗	✓	1M	5	1.8K
OmniNOCS	✓	✓	380k	97	> 450k

Table 1: Comparison to other NOCS datasets: Other existing NOCS datasets are limited in number of classes and instances. Note that Wild6D has 1M images, but these are used for self-supervised training, as it does not include NOCS ground truth (GT).

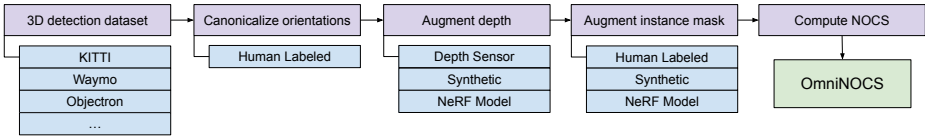

Fig. 2: Preparation of the OmniNOCS dataset: First, we ensure that object orientations are canonical across different all subsets of OmniNOCS. Next, we re-compute depth for datasets where depth is not available or is too noisy. Finally, we annotate objects with (pseudo) instance mask labels where ground truth masks are not available.

3 OmniNOCS dataset

3.1 Data statistics

We create OmniNOCS, a first-of-its-kind *large and diverse* NOCS dataset comprising data from several classes and domains. OmniNOCS uses data from self-driving scenes [9, 15, 19, 46], indoor scenes [5, 44, 45, 49], and object-centric videos [1]. Each of these sources use cameras with different parameters, ranging from phone cameras to wide FOV cameras mounted on self-driving cars. The number of object instances per image also varies widely, from single-object images [1] to images with hundreds of objects [44, 46].

OmniNOCS has 97 object categories across 380k images. This by far exceeds the diversity of existing NOCS datasets (with less than 6 categories as shown in Table 1). The number of instances exceeds that of other NOCS datasets by more than 2 orders of magnitude. It also contains more images with NOCS annotations (380k). Note that while Wild6D has 1M images, these do not have ground truth NOCS or pose annotations, and includes every frame of object-centric videos. OmniNOCS is also more diverse than the most diverse 3D detection dataset (Omni3D [7] with 98 categories). 307k images in OmniNOCS are from real world scenes and 73k are synthetic. 107k of the images in OmniNOCS are from outdoor scenes, 102k images are from complex indoor environments, and 140k are from object-centric videos.

3.2 Data preparation

Computing the NOCS for objects in an image requires canonically oriented 3D object bounding boxes, depth, and instance segmentations. Since the datasets we use were originally only proposed for monocular 3D detection, many of them lack dense depth or instance segmentation annotations (highlighted entries in Table 2). Additionally, NOCS

3D Dataset	Depth source	Instance mask source	#Images	#Classes
KITTI [19]	LiDAR	Human-labelled [23]	7.4k	7
SUN-RGBD [45]	Depth Camera	Segmentation model [25]	10k	72
Objectron [1]	NeRF	NeRF + Human label	132k	9
nuScenes [10]	LiDAR	Segmentation Model [14]	30k	9
Hypersim [44]	Synthetic	Synthetic	64k	31
ARKitScenes [5]	Depth Camera	Segmentation model [25]	60k	15
Cityscapes 3D [15]	Stereo	Human-labelled	3.4k	8
Virtual KITTI [9]	Synthetic	Synthetic	4.1k	3
NOCS-Real275 [49]	Depth camera	Human-labelled	7k	6
Waymo OD [46]	LiDAR	Human-labelled	62k	7
OmniNOCS			**380k**	**97**

Table 2: Constituent 3D detection datasets used in OmniNOCS: We augment many of these datasets with depth and masks, as they are missing in the original data (highlighted entries). In addition, we canonicalize the orientations of bounding boxes across all datasets. #Classes lists the number of classes we use from each dataset. Those above the dashed line are part of Omni3D [7].

also requires all instances within a category to have consistent orientation annotations. While this is certainly not true *across* different 3D detection datasets, it is also not true *within* some large synthetic datasets such as Hypersim. OmniNOCS aggregates several datasets, with the addition of 1) new depth estimated via Mip-NeRF reconstructions, 2) additional instance masks via segmentation models and human labels, 3) manual labelling of coordinate axes to give consistent object-centric coordinate frames. This enables NOCS to be computed for every image in the dataset. This multi-stage process for OmniNOCS creation is illustrated in Figure 2 and explained below. A summary of the resulting OmniNOCS dataset is provided in Table 2.

Orientation canonicalization: Although the datasets we use contain oriented 3D bounding boxes, they vary in their level of canonicalization. Some datasets have their class-canonical orientations i.e, all instances of a particular class are oriented consistently *within the dataset* (for example, all cars in [19] have X axis pointing forward, and Z upwards). In such cases, we ensure that this canonicalization is consistent with all of OmniNOCS by applying a fixed class-specific offset orientation for the dataset. Datasets like Hypersim [44] have no class canonicalization at all, iė, although objects have tight bounding boxes, their XYZ axes directions are different for each instance. We manually label each object in such datasets to select the canonical orientation out of six possible orientations for the bounding box. More details on the labelling process are provided in the supplementary material. Some classes may have more than one choice for canonical orientations (due to symmetry), in which case an orientation is selected arbitrarily.

Depth augmentation: For outdoor datasets, we use sparse depth from LiDAR if available. We recomputed the depth on Cityscapes using a state-of-the-art stereo depth model [55], as we found the depth from the original dataset to be noisy. Since Objectron [1] does not contain dense depth, we train Neural Radiance Fields [4, 37] for each video sequence in the dataset to obtain dense depth.

Instance segmentation: Since many datasets we use were intended for 3D detection, they do not camera instance segmentations. We annotate ARKitScenes and SUNRGBD objects using instance masks from Segment-Anything (SAM) [25], prompting it with the

projected 3D bounding box. Although SUN-RGBD provides segmentation masks, we find that the SAM annotations are of superior quality. For Objectron, we create accurate masks by efficiently annotating in 3D space. Specifically, we apply the pipeline of [4] to each of the Objectron videos. We fuse the resulting NeRF depth-maps to create a mesh. We post-process the 3D mesh, and get rid of the redundant vertices while keeping the vertices of the object. Finally, we create the masks for multiple frames by measuring the distance between each pixel on the depth map and the object mesh: pixels whose depth is far from the surface of the mesh are discarded, a process which also handles occlusions.
NOCS computation: As proposed in [49], the NOCS coordinate x_{nocs} for a point on the object surface is defined as:

$$x_{nocs} = \frac{1}{s} \, {}^{obj}T_{cam} \; x_{cam} \tag{1}$$

where ${}^{obj}T_{cam} = [{}^{obj}R_{cam} | {}^{obj}t_{cam}]$ is the 6DoF transformation from the camera to the object frame, x_{cam} the 3D location of the point in the camera frame, and s is a scalar, the size of the diagonal of the object's tight bounding box. It can be interpreted as the object shape scaled to a box with a unit diagonal. For each image, starting from a 3D pointcloud (x_{cam}) obtained from the depth, we collect points on each object (x^i_{cam}) using its 3D bounding box and instance mask, which can be transformed to the NOCS coordinate (x^i_{nocs}) using (1). We store NOCS as a 3-channel 2D map (i.e x^i_{nocs} at its corresponding 2D location obtained by projecting x^i_{cam}). Our final result is the OmniNOCS dataset which contains instance segmentations, NOCS maps, and 3D bounding boxes for objects across 97 classes.

4 NOCSformer model

We propose a novel architecture for monocular NOCS prediction termed "NOCSformer" trained on OmniNOCS. NOCSformer primarily uses self-attention layers, and is the largest trained monocular NOCS prediction model to date. NOCSformer also contains a 3D size head and a learned PnP head that are used to estimate a canonically oriented 3D object bounding box from the NOCS.

4.1 Model architecture

As shown in Figure 3, the NOCSformer architecture comprises an image backbone, a NOCS head, a size head, and a learned PnP head.
Backbone: Our backbone is a frozen Vision Transformer (ViT) [17], that uses DINOv2 weights [40] from self-supervised pretraining. This choice is motivated by recent findings on using frozen DINOv2 backbones for understanding depth, multi-view correspondences, or relative pose [20, 56]. Additionally, we use a Dense Prediction Transformer (DPT) [43] architecture to fuse low-resolution DINOv2 representations from multiple intermediate layers and upsample them by a factor of 8. This higher feature resolution is desirable to improve predictions on smaller objects. We train the DPT layers while keeping the ViT layers frozen. We use 2D input boxes to sample the DPT features using

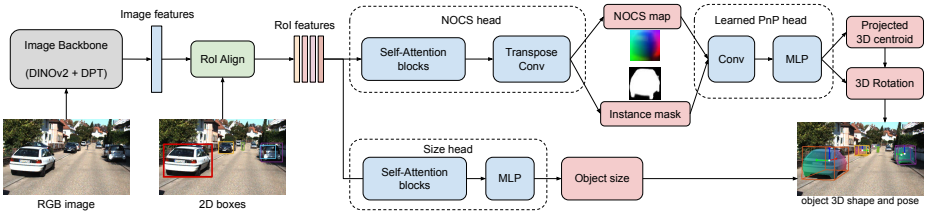

Fig. 3: Architecture for NOCSformer: We use a transformer backbone to extract features from the input image, pool them using the 2D box RoIs, and feed the per-RoI features to the NOCS and size heads. Our novel NOCS head jointly predicts the NOCS and instance mask for the RoI. Our learned PnP head for pose estimation uses the predicted NOCS and instance mask to predict the projected 3D coordinate and 3D rotation of the object.

RoIAlign [22] into a 28x28 fixed resolution grid that is fed to the NOCS and size heads. While DPT has been previously leveraged for tasks that provide dense supervision, such as semantic segmentation and depth prediction, we use it in an instance-level prediction context, training it using supervision from object RoIs.

NOCS head: All existing NOCS prediction models [31, 48, 49] use MaskRCNN heads [22] with convolutional layers, in some cases with a separate head for each NOCS coordinate [49]. While this works on small datasets, we find that it significantly limits performance on OmniNOCS. NOCSformer therefore uses a simple (but large) transformer NOCS head with 10 self-attention blocks and one final upsampling conv layer to jointly predict both the instance mask and the NOCS map in the same output head. Similar to [49], for each NOCS coordinate, we predict logits over non-overlapping bins, and use the softmax to obtain the final NOCS value. Our ablations show that this choice works best on our challenging OmniNOCS dataset.

Size and learned PnP heads: We use a size head to predict the 3D size to scale the NOCS predictions to metric object coordinates. Our size head also uses self-attention blocks. Contrary to previous work [7,27,49], it does not use any class-specific layers or per-class average size statistics, as these limit scalability to larger and diverse vocabularies. Our learned PnP head is inspired from [31,48] and uses convolutional layers to predict the 3D orientation and a 2D projection of the 3D centroid. The orientation is predicted using the 6D partial rotation matrix representation [57] in allocentric space [7,27].

4.2 Localization from NOCS predictions

To localize the object in 3D, we first scale the predicted NOCS using the predicted size to obtain unnormalized metric object coordinates. NOCSformer's learned PnP head predicts the object orientation in allocentric space, which is converted to the egocentric frame using the predicted projection of the 3D centroid. Using the unnormalized 3D object coordinates, object orientation, and the 3D centroid ray, the object depth is estimated from the corresponding 2D points.

Alternatively, the 6DoF pose for the object can be solved directly (without using the learned PnP head) by solving a PnP problem using efficient solvers [29]. However, the orientations so obtained are known to be more sensitive to errors in NOCS coordinates, and less robust when compared to learned methods [31,48].

4.3 Losses

When training NOCSformer, we supervise the NOCS, mask, 3D size, orientation and centroid predictions using appropriate losses. We supervise predicted masks using a simple L2 loss with respect to the ground truth instance mask. Since our NOCS head uses binned prediction where the final NOCS value is a softmax over non-overlapping bin logits, we use a combination of cross-entropy and regression losses.

$$\mathcal{L}_{mask} = ||\texttt{mask}_{pred} - \texttt{mask}_{gt}||_2$$
$$\mathcal{L}_{NOCS} = \texttt{softmaxCE}(\hat{\mathbf{n}}, \hat{\mathbf{n}}_{gt}) + ||\mathbf{n} - \mathbf{n}_{gt}||_1$$

where $\hat{\mathbf{n}}$ are the predicted logits over the discretized bins and $\mathbf{n}$ the continuous NOCS coordinate prediction.

In addition to the supervised loss above, we also use a variant of the self-supervised NOCS loss $\mathcal{L}_{ss}$ from [11] that minimizes the reprojection error of the predicted NOCS using the ground truth pose and predicted mask. More details about $\mathcal{L}_{ss}$ are in the supplementary material. The NOCS and mask losses are applied to the fixed resolution grid predictions. Since our NOCS ground truth can be sparse, it is applied at only those locations on the grid that have valid NOCS ground truth.

Since our 3D size head also uses binned prediction, we use a combination of softmax cross-entropy and L1 loss for supervision. Note that our size loss is normalized by the ground truth size, in order to penalize errors on smaller objects equally. For the learned PnP head, we also supervise the centroid and rotation predictions using L1 losses in their output space.

$$\mathcal{L}_{size} = \texttt{softmaxCE}(\hat{\mathbf{s}}, \hat{\mathbf{s}}_{gt}) + |\mathbf{s} - \mathbf{s_{gt}}|/\mathbf{s_{gt}}$$
$$\mathcal{L}_{rot} = \|^{c}\mathbf{R}_{o_{pred}} - {}^{c}\mathbf{R}_{o_{gt}}\|_{1,1}$$
$$\mathcal{L}_{centroid} = ||\mathbf{c}_o - \mathbf{c}_{o_{gt}}||_2$$
$$\mathcal{L}_{PnP} = \mathcal{L}_{rot} + \mathcal{L}_{centroid}$$

The total loss for training NOCSformer is a weighted sum:

$$\mathcal{L}_{total} = w_{size}\mathcal{L}_{size} + w_{mask}\mathcal{L}_{mask} + w_{NOCS}\mathcal{L}_{NOCS} + w_{ss}\mathcal{L}_{ss} + w_{PnP}\mathcal{L}_{PnP}$$

5 Experiments

Although previous works predict NOCS accurately on a few categories [31, 48, 49], they only evaluate on 3D detection (localization) or pose estimation tasks, without quantifying the accuracy of the predicted NOCS. In section 5.1, we propose metrics and establish a benchmark to evaluate NOCS on the OmniNOCS dataset. We compare the localization accuracy of NOCSformer against existing benchmarks, by evaluating its localization accuracy on nuScenes [10] in section 5.2. We also evaluate the unique ability of NOCSformer to transfer to unseen datasets and domains. Finally, we provide ablations on critical design choices made in our model in section 5.4.

Fig. 4: Example results of our single unified NOCSformer model across various datasets and object classes. The left column shows input images and query 2D bounding boxes. The center column shows the NOCS+instance maps predicted by NOCSformer along with the estimated 3D pose overlaid on the input image. The NOCS can be used with the 3D boxes and object size to lift the objects to a 3D pointcloud, which is shown in the right column. Last row contains two examples.

Datasets: For our experiments, we train NOCSformer on the OmniNOCS dataset containing 97 classes, holding out images from the NOCS-Real275 dataset. We use NOCS-Real275 to evaluate NOCSformer's cross-dataset generalization capabilities by not training on it. NOCS-Real275 has 6 classes that overlap with the rest of OmniNOCS, although they are not the top 20 classes. It also differs from the rest of OmniNOCS in terms of camera parameters and the context of objects in the scene.

5.1 NOCS prediction accuracy

We propose the use of NOCS mAE and NOCS mPSNR to evaluate the accuracy of predicted NOCS. For each object the mean Absolute Error (mAE) and mPSNR are computed for all points in the intersection of ground truth and predicted masks, and reported as a mean of per-category means. However, the NOCS mAE/mPSNR do not penalize undersegmentations / sparse predictions. Since we also would like the NOCS predictions to span the full visible instance, we also evaluate the 2D mask mIoU.

We evaluate NOCSformer on a subset of OmniNOCS that contains 75 classes with accurate ground truth NOCS and masks, see Table 3. It is able to predict NOCS with errors less than 9% (0.089 mAE on OmniNOCS). We also evaluate on the held-out NOCS-Real275 dataset, which was not used to train NOCSformer. We find that the zero-shot NOCSformer outperforms the NOCS baseline from [49] that was fully trained on this dataset, a strong indication of the generalization capabilities of NOCSformer.

5.2 3D localization accuracy

NOCS provides dense 3D-2D correspondences that can be used to estimate the 3D object oriented bounding box. This is done by using the predicted object size and solving a PnP problem, as explained in Sec. 4.2. We evaluate the accuracy of our estimated bounding boxes in different settings:

Outdoor scenes: We compare NOCSformer's 3D localization accuracy to that of other 3D detection models using the nuScenes true positive localization metrics on the challenging nuScenes dataset (included in OmniNOCS). We compare to Cube R-CNN, since it is the only other model that generalizes across diverse datasets. However, Cube R-CNN has two notable differences to NOCSformer: 1) being a detection model, it also jointly detects 2D object regions, and 2) it uses Chamfer loss causing it to have high orientation errors. As a more comparable baseline, we use a variant of our model (termed "Cubeformer"), by replacing our NOCS head with the cube head from Cube R-CNN

	NOCS-Real275			OmniNOCS		
Method	NOCS MAE↓	NOCS PSNR↑	Mask IoU↑	NOCS MAE↓	NOCS PSNR↑	Mask IoU↑
NOCS baseline [49]	0.121	16.345	86.10	-	-	-
NOCSformer	**0.107**	**18.527**	**89.03**	**0.094**	**20.245**	**78.50**

Table 3: NOCS and mask prediction evaluation: We report metrics for NOCSformer on 75 classes of OmniNOCS: it is the first model that is capable on predicting NOCS on such diverse data. Additonally, NOCSformer outperforms [49] which trained on NOCS-Real275, without training on NOCS-Real275.

Method	Multi-dataset	mATE (m)↓	mAOE (rad)↓	mASE (%)↓	mIoU↑
FCOS3D [51]	✗	0.777	0.400	**0.231**	-
PGD [50]	✗	0.675	0.399	0.236	-
EProPNP [12]	✗	0.676	0.363	0.263	-
EProPNP + TTA [12]	✗	**0.653**	**0.319**	0.255	-
Cube R-CNN [7]	✓	**0.650**	1.305	**0.283**	0.349
Cubeformer	✓	0.790	0.573	0.301	0.280
NOCSformer	✓	0.887	**0.558**	0.291	**0.377**

Table 4: Outdoor localization: Comparison of 3D localization accuracy on the nuScenes subset of OmniNOCS. Top half shows methods that are only trained on nuScenes: while these perform better on nuScenes itself, they do not generalize to other datasets and classes. OmniNOCS is competitive with Cube R-CNN on mIoU while also predicting canonical orientations and object coordinates. Note that mIoU and orientation error only uses the predicted yaw orientation component.

without a Chamfer loss. It uses the input 2D boxes. More details about Cubeformer are in the supplementary material. From Table 4, we find that NOCSformer's box orientations are canonical and more accurate than both Cube R-CNN and Cubeformer. Moreover, the boxes estimated from NOCSformer's NOCS predictions are comparable in translation and scale errors to those of Cube R-CNN and Cubeformer, even though it does not directly regress depth. While all 3 methods generalize across several classes and domains, baselines trained solely on nuScenes localization (top half) significantly outperform them on this task.

Cross-dataset generalization (indoor): Here, we hold out NOCS-Real275 from OmniNOCS when training NOCSformer, and use it evaluate NOCSformer's ability to generalize to unseen domains. NOCS-Real275 features a tabletop setting with multiple objects, which is semantically different from our other indoor datasets. We compare against Cube R-CNN [7] and the NOCS [49] model that was trained from scratch on this dataset alone. [49] also uses class-specific model parameters and additional losses for symmetric objects that do not scale to larger vocabularies.

The results are shown in Table 5. We find that NOCSformer is more accurate at transferring to this unseen dataset compared to Cube R-CNN, indicating the generalizability of NOCS-based localization methods over that of box-regression methods. Note that the mAP metrics used in the NOCS-Real275 datasets may also be affected by the false positives/negatives from Cube R-CNN. However, both models are worse that the supervised NOCS baseline.

			mAP	
Method	Transferred?	Depth input?	3D IoU @ 25	3D IoU @ 50
NOCS model [49]	✗	✓	**79.6**	**72.4**
CubeRCNN [7]	✓	✗	14.9	4.1
NOCSformer	✓	✗	**43.5**	**10.6**

Table 5: Cross-dataset generalization: Comparison of localization accuracy on NOCS-Real275 for all classes, using mAP at different thresholds. This dataset has been held out when training CubeRCNN and NOCSformer. NOCSformer is able to generalize to NOCS-Real275 *without any additional training*.

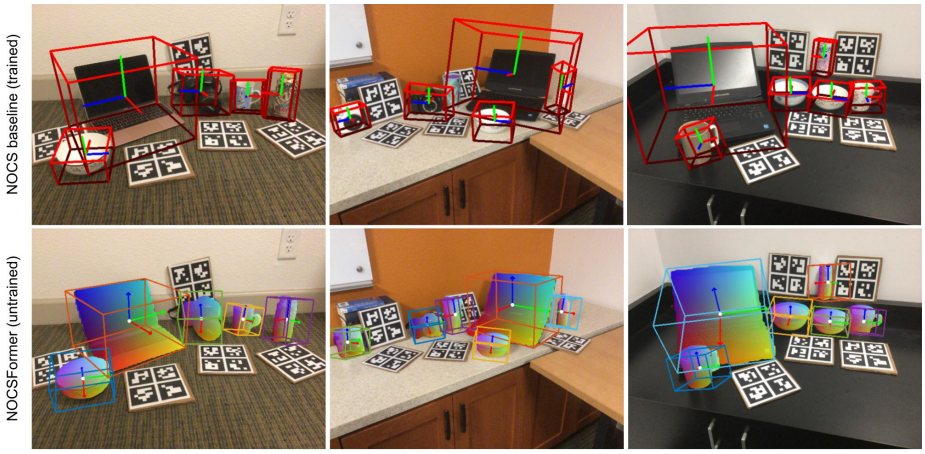

Fig. 5: Generalization across datasets: NOCSformer can generalize to new datasets that present new camera models and object domains. We show this zero-shot capability of NOCSformer (bottom row) on the NOCS-Real275 test set without training on NOCS-Real275 dataset. The predictions from a NOCS baseline [49] trained explicitly on the NOCS-Real275 dataset are shown in the top row.

5.3 3D orientation accuracy

A key challenge for 3D localization models trained on multiple datasets [7] is predicting canonical object orientations, as these datasets typically differ in their object pose conventions. OmniNOCS ensures that object poses in all constituent datasets are consistent within a category, enabling our model to predict canonical orientations. We evaluate this by computing the accuracy at different orientation error threshold on KITTI-val [13] in Table 6. We compare the accuracy of our orientation predictions to those of Cube R-CNN [7], which are trained using Chamfer loss, and therefore suffers at predicting canonical orientations. We also compare against our Cubeformer baseline that directly supervises the orientations. We find that orientations estimated using NOCSformer are more consistent and accurate than both our baselines.

	Gravity axis			Heading axis (X)		
Method	Accuracy @ 1 deg	Accuracy @ 5 deg	Accuracy @ 90 deg	Accuracy 5 deg	Accuracy @ 10 deg	Accuracy @ 90 deg
CubeRCNN	5.75	18.29	51.74	19.43	21.55	34.17
Cubeformer	80.79	98.52	100.0	41.99	55.88	77.26
NOCSformer	**84.24**	**99.95**	**100.0**	**49.16**	**59.65**	**81.99**

Table 6: 3D orientation accuracy for models trained on multiple datasets: Models that use Chamfer distance (CubeRCNN) for supervising orientation heads end up being inconsistent in their orientation predictions. The results are averaged over 5 classes in the KITTI-val subset [13].

Architecture	NOCS PSNR↑	Mask IoU↑
NOCSformer	**20.431**	**81.69**
Hourglass head	-3.064	-3.16
MaskRCNN head	-5.23	-21.27
w/o discretized prediction	-4.11	-

Table 7: Comparison of different architecture choices for the NOCS and mask prediction head: We experiment with transformer, hourglass and MaskRCNN architectures. We compare whether discretized prediction of NOCS is better than continuous regression for transformer heads. We quantify the difference in performance when using separate heads for mask and NOCS prediction as opposed to a single head.

5.4 Ablations

NOCS head architecture: While all existing architectures use a few CNN layers for their NOCS heads [49], we find that this significantly limits the NOCS prediction performance when scaling to more classes. Using our transformer NOCS head provides a 5.23dB improvement in NOCS PSNR and a 21.27% improvement in mask mIoU. We also experiment with a larger convolutional head: the Hourglass model from [6]. We find that the NOCSformer NOCS heads are even better than Hourglass, with an improvement of 3.06dB on NOCS PSNR and 3.16% on mask IoU (as shown in Table 7).

Discretized versus continuous prediction of NOCS: NOCSformer models the final NOCS prediction as a classification layer with 50 non-overlapping bins. This was observed to be better than a linear regression layer with a MaskRCNN head in [49]. We find that it is also significantly better when using a transformer head, improving the NOCS prediction PSNR by 4.11 dB, as in Table 7.

6 Conclusion and Future work

This paper has introduced a new large scale dataset of Normalized Object Coordinates (NOCS) for a wide variety of object classes in indoor and outdoor scenes. It has also proposed a single transformer-based model NOCSformer that can predict object 3D pose, size, and NOCS for any of these objects given 2D bounding box inputs. These allows our model to obtain 3D shape and oriented bounding boxes of objects in metric scale from a single input image. This represents the first attempt to generalize NOCS estimation beyond small datasets of narrow domain, increasing the number of object categories available by an order of magnitude. We hope this provides a means for others to explore large-scale monocular 3D object pose and shape estimation.

Some limitations of our current method include the handling of symmetric objects, where the coordinate system has multiple possible solutions, and reflected geometry, such as left / right shoes. Future work could address these issues, for example, minimizing over multiple coordinate frames in the loss for symmetric objects, and potential estimation of left / right coordinate frames for reflected geometry.

Appendix

This appendix provides additional qualitative results, including results on in-the-wild internet image collections. Detailed OmniNOCS statistics and more information about the annotation process are provided in Section B. Section C contains more details on NOCSformer architecture and training.

Fig. 6: NOCSformer trained on OmniNOCS dataset generalizes to web **stock images** that are outside the training dataset. The predicted NOCS and 3D bounding boxes are shown on right for each input image and 2D query. These results demonstrate the capability of NOCSformer trained on OmniNOCS for 3D object reconstruction of in-the-wild images.

A Qualitative results of NOCSformer

We present additional qualitative results for NOCSformer trained in different settings – outdoor scenes (Fig. 8) and indoor scenes, (Fig. 9). We also provide results on in-the-wild images from the web (Fig. 6) and some images from the COCO dataset (Fig. 7).

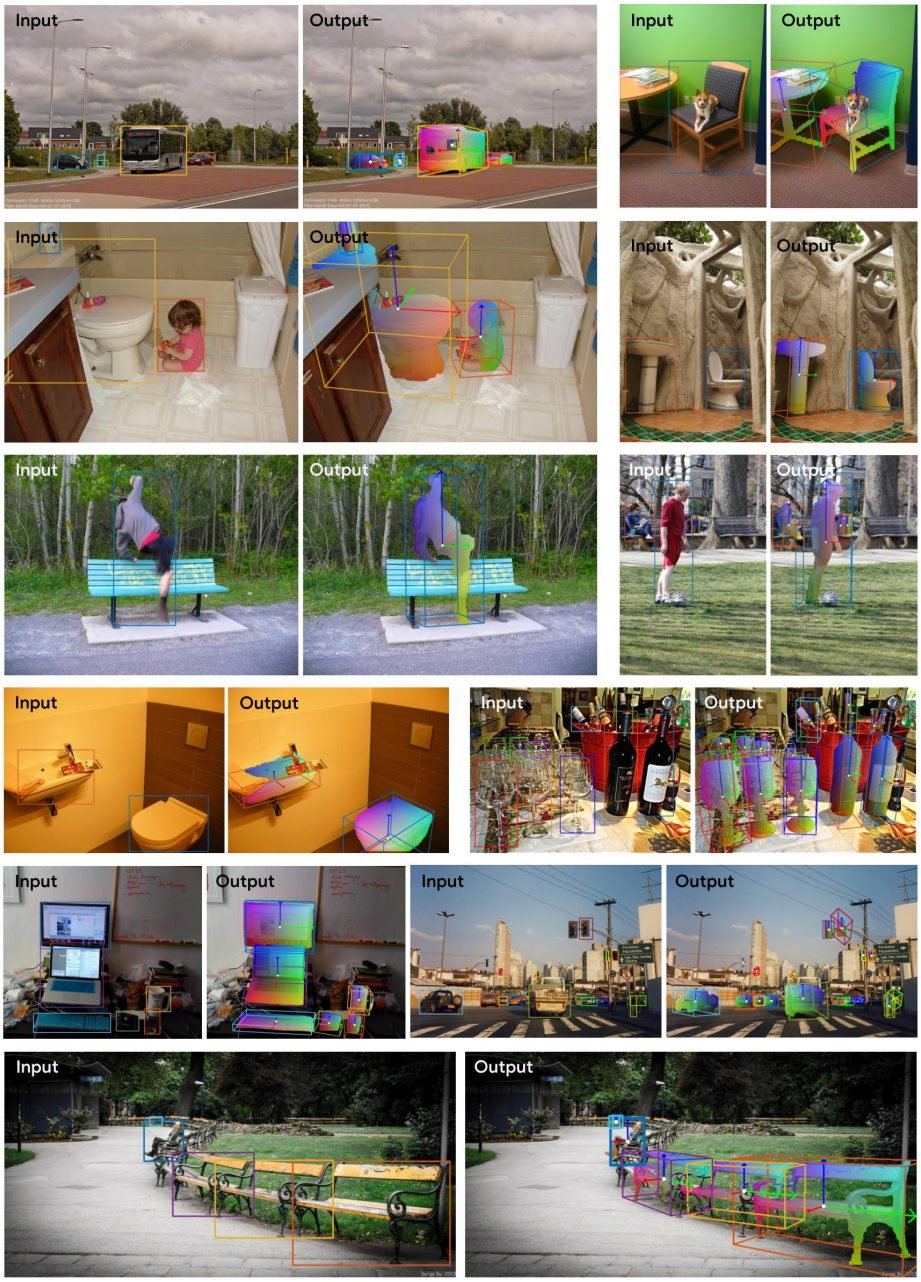

Fig. 7: Results of NOCSformer (trained on OmniNOCS dataset) on images from the **COCO dataset**. For each image pair, the left image shows the input 2D bounding box, and the right image shows the predicted NOCS and 3D bounding box. Note that COCO is **not** part of OmniNOCS. However, the model generalizes to the challenging COCO images, predicting NOCS and 3D bounding boxes from 2D queries. The model is also able to generalize to unseen (though related) classes – for example generalizing to park benches when only trained on sofas and chairs.

Fig. 8: Qualitative results of **NOCSformer** model for various outdoor datasets. Left column shows the input images and query 2D bounding boxes. The center image shows the predicted per object NOCS, segmentation, and 3D oriented bounding box from our model corresponding to each input query overlaid on the input image. The right image shows the object coordinates lifted to 3D using the predicted 6DoF pose.

Fig. 9: Qualitative results of **NOCSformer** model for various indoor datasets. Left column shows the input images and query 2D bounding boxes. The center image shows the predicted per object NOCS, segmentation, and 3D oriented bounding box from our model corresponding to each input query overlaid on the input image. The right image shows the object coordinates lifted to 3D using the predicted 6DoF pose.

A.1 Generalization to in-the-wild images

Fig. 6 shows NOCS predictions on online stock images, highlighting the model's ability to generalize beyond the training contexts. Additionally, we run our model on the COCO dataset [32], and show results in Fig. 7. The results confirm these findings on the model's generalization capabilities. Since our model does not contain any class specific parameters, it can also be queried on classes it is not trained on. In such cases, it tends to perform reliably for classes that are closely related to the training classes. For example, it performs well on park benches in Fig. 7 despite only being trained on couches, sofas and chairs.

A.2 Additional results on OmniNOCS

Fig. 8 shows results from KITTI [19], Cityscapes [15], Waymo [46], and nuScenes [10] for different object classes including cars, bikes, pedestrians, and trucks. Our method is able to predict NOCS and 3D bounding boxes reliably even under occlusions, during the night time, or in severe rainy conditions. We use the NOCS and 3D bounding box to lift the object to a 3D pointcloud. The rightmost 3D column shows that the obtained pointcloud respects relative 3D distances and orientations. Fig. 9 shows results from Objectron [1], Hypersim [44] and SUN-RGBD [45] in indoor environments, taken from different types of cameras. This also includes results on some less frequent classes such as bookshelves, curtains, and mirrors.

B OmniNOCS dataset details

B.1 Data statistics

OmniNOCS has more than 2.2M object instances spanning the train, val and test splits (not counting repeated instances). A histogram of the number of instances in the top and bottom 50 categories is shown in Fig. 10.

Fig. 11 provides more insights into the constitution of OmniNOCS. We use images from 10 other 3D detection datasets, which vary in terms of the number of instances, number of categories, and the scene complexity (measured by number of instances per image). While Hypersim provides the greatset number of objects, SUN-RGBD contributes the largest number of categories. The most complex scenes are from Hypersim (indoors) or KITTI and Waymo (outdoors). We show more examples for diverse object categories from the OmniNOCS dataset in Fig. 13.

B.2 Canonical orientation labeling

To produce NOCS that are consistent across a category in OmniNOCS, it is required to have bounding boxes with consistently oriented axes. For example, all chairs in OmniNOCS have the Z-axis pointing upwards, and X axis pointing forwards. The definition of the canonical orientation is class-dependent. For some classes this is fully defined by geometry and the direction of gravity. For example, we use the longest edge of a book as its Z axis, and the shortest edge as the X axis. For windows, we use the

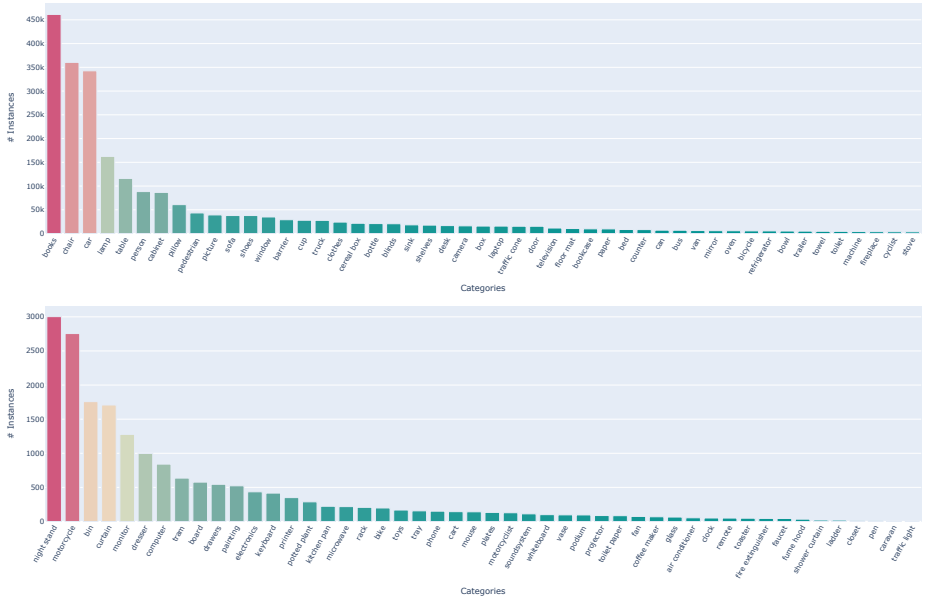

Fig. 10: A histogram of the number of instances for each category in OmniNOCS: (top) most frequent categories (bottom) least frequent categories.

upward direction as the Z axis and the shortest edge as the X axis, facing the direction of the camera. For other categories, such as chairs, desks, vehicles, pedestrians, these are defined based on the object semantics, and these need to be manually labeled. For example, most people would agree on what the front of a chair is, even though this might not be clear from the bounding box dimensions alone. We choose to label this front direction as the X axis of the chair, and make sure to do so consistently across the entire dataset.

For the constituent datasets of Omni3D, the existing bounding box labelling is typically inconsistent between datasets (inter-dataset inconsistency). It can also be inconsistent within a dataset (intra-dataset inconsistency). We explain how we resolve this in each case below:

Intra-dataset Inconsistency: This means that orientations for different instances of the same category can be inconsistent, within a single dataset. This happens in the case of Hypersim – the orientation axes for a bounding box are chosen based on instance geometry, and can vary across instances of the same category (see Fig. 12). For each instance, we manually choose an offset rotation that makes the resulting orientation consistent across the category, and with the rest of OmniNOCS.

Inter-dataset Inconsistency: In this more common case, orientations for all instances of a category are consistent, but only within the smaller dataset. For example, all chairs in Objectron and SUN-RGBD are oriented consistently within each dataset, but they are not consistent with each other. In this case, we apply the same offset rotation to all instances from a particular dataset to ensure that they are consistent with the rest of OmniNOCS.

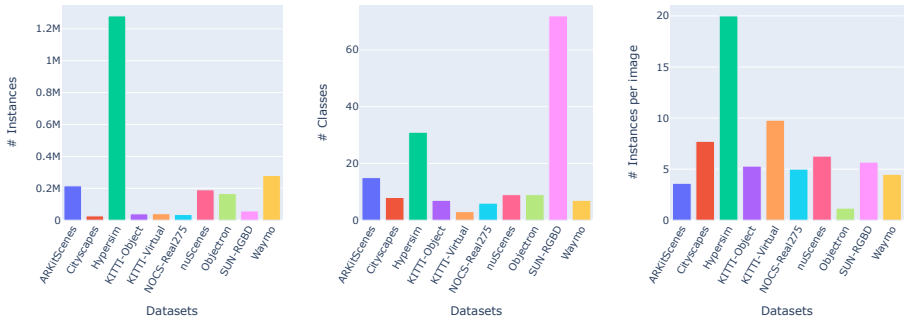

Fig. 11: Statistics for the datasets that are used in OmniNOCS. These datasets span different domains and classes. They also also differ significantly in the number of instances per image.

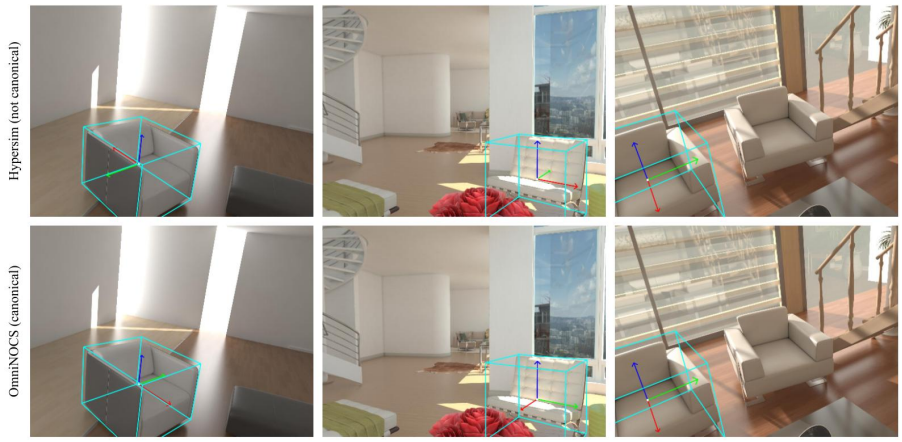

Fig. 12: Instance-level orientation canonicalization for Hypersim: Each chair instance in Hypersim has its X (red) and Y (green) axes chosen differently. To produce consistent NOCS coordinates, we apply an offset rotation to the original orientations such that the resulting orientations are consistent across all instances of the class. We find this offset by manually inspecting all Hypersim objects.

Fig. 13: Example annotations from our OmniNOCS dataset: Each frame contains multiple objects, with each object having a 3D bounding box with canonical orientations, NOCS coordinates and instance mask.

C Model details

We discuss the details of our implementation of NOCSformer and Cubeformer, including the data processing, augmentation, model architecture details, and training regime in the following sections.

C.1 Data augmentation

We augment our training data with resizing by scaling the images randomly in the range [0.5, 1.5]. We also scale the camera intrinsics accordingly to ensure the 3D ground truth remains correct after augmentation.

C.2 Model architecture

Backbone: We use the DINOv2 B-14 [40] transformer as our image backbone, with a higher input image resolution of 896×896. We also use DPT layers [43] that fuse features from every third DINOv2 self-attention block, and upsamples the final features by a factor of 8 with a feature dimension of 512.
Cubeformer: While NOCSformer regresses object coordinates, models like Cube R-CNN [7] directly regress bounding box parameters: the projected 3D centroid, 3D size, 3DoF orientation, and depth. While a direct comparison of the localization accuracy of NOCSformer to Cube R-CNN is provided in Table 4, this is not a fair comparison because Cube R-CNN uses a different convolutional backbone that is trained from scratch on the target dataset for both 2D region proposal and 3D localization. In contrast, NOCSformer is a localization model that accepts 2D bounding box inputs. We therefore design a baseline we term *Cubeformer*, that uses the same image backbone as NOCSformer, input 2D bounding boxes, and predicts 3D bounding boxes using a self-attention based cube head unlike the convolutional head of [7]. While Cubeformer can be supervised using Chamfer losses like Cube R-CNN [7], this causes the predicted orientations to be inconsistent. We therefore use direct L1 or L2 losses in the output parameter space (6D orientation vector, 2D projected centroid, scalar depth, and 3D size) as supervision. Our Cubeformer baseline therefore has a much better mAOE compared to Cube R-CNN in Table 4. However, our NOCSformer's orientations are better than both Cube R-CNN and Cubeformer.

C.3 Self-supervised reprojection error

As explained in Section 4.3, we use the NOCS reprojection error as self-supervision for training NOCSformer. The loss is inspired from [11] but unlike them, we do not require the uncertainty modelling / KL divergence step. We consider a 3D NOCS prediction $\mathbf{n}$ corresponding to a 2D pixel with image coordinates $\mathbf{p}$. The ground truth orientation is ${}^{c}\mathbf{R}_{o_{gt}}$, translation ${}^{c}\mathbf{t}_{o_{gt}}$ and scale $\mathbf{s}_{gt}$. We obtain 3D object coordinates, first in object frame, ${}^{o}\mathbf{x}$, then in camera frame ${}^{c}\mathbf{x}$, and then project to the image to obtain the reprojected NOCS point $\mathbf{p}_{proj}$

$$
\begin{aligned}
{}^{o}\mathbf{x} &= \mathbf{s}_{gt}.\mathbf{n} \\
{}^{c}\mathbf{x} &= {}^{c}\mathbf{R}_{o_{gt}}\,{}^{o}\mathbf{x} + {}^{c}\mathbf{t}_{o_{gt}} \\
\tilde{\mathbf{p}}_{proj} &= \mathbf{K}_{c}\,{}^{c}\mathbf{x}
\end{aligned}
$$

where . denotes element-wise scalar multiplication, $\mathbf{K}_c$ is the camera intrinsic matrix and $\tilde{\mathbf{p}}_{proj}$ is the homogeneous form of the projected point $\mathbf{p}_{proj}$. The self-supervised loss is given by:

$$
\mathcal{L}_{ss} = \begin{cases} ||\mathbf{p} - \mathbf{p}_{proj}||_2, & \text{mask}_p = 1 \\ 0, & \text{otherwise} \end{cases}
$$

where mask_p is the predicted instance mask at $\mathbf{p}$. Note that we do not use gradients from this loss to supervise the predicted mask. The loss is termed self-supervised, as it does not need a ground truth NOCS map, but it requires the ground truth pose and size labels.

C.4 Training

We train our models using the Adam optimizer with a base learning rate of 1e-4 for 200k steps. We use a linear warmup of the learning rate over the first 1000 steps. We use a weight decay of 1e-6 for the weights of the convolutional layers and 1e-4 for the MLP layers. We use a dropout of 20 % for the MLPs. We clip the global gradient norm to 10.0. We use a batch size of 128, divided amongst 16 TPU cores (or 16 A100 GPUs). The models take approximately 40 hours to train.

C.5 Per-category NOCS quality

Section 5.1 in the paper quantitatively evaluate the quality of NOCS produced by the model using the mAE, mPSNR and mIoU metrics. The numbers for these metrics in Table 3 are averaged over 75 classes in OmniNOCS that have high quality NOCS ground truth. Here, we provide the per-class numbers in Table 8 to analyze the variability of predictions among classes. In general, we see that categories that are either rare or very diverse have higher NOCS errors compared to other categories. For example, toys and projectors have PSNR of 17.92 and 12.21 respectively, whereas cars have a PSNR of 27.45.

D Temporal consistency results

We provide a video of NOCSformer's independent predictions on each frame for some sequences from Objectron [1], attached in the supplementary material. The independent NOCS and pose estimates are consistent temporally, without use of any smoothing/filtering techniques.

Class	car	blinds	van	monitor	curtain	mirror	toilet	air conditioner	closet	stove	cyclist	board	clothes	pedestrian	toaster	dresser	painting	bookcase	shelves
MAE	0.037	0.038	0.041	0.041	0.045	0.047	0.047	0.05	0.054	0.056	0.056	0.057	0.058	0.058	0.059	0.061	0.061	0.062	0.062
PSNR	27.45	27.92	26.07	26.77	26.14	25.97	24.31	25.97	22.68	22.86	22.98	26.0	21.19	23.8	22.45	23.74	23.53	23.03	23.24
IoU	0.84	0.943	0.831	0.814	0.874	0.758	0.953	0.845	0.974	0.95	-	0.937	0.227	0.659	0.958	0.908	0.884	0.853	0.675

Class	door	truck	bottle	printer	sofa	computer	fireplace	picture	chair	vase	person	coffee maker	sink	potted plant	drawers	television	bed	keyboard	microwave
MAE	0.064	0.065	0.066	0.066	0.067	0.068	0.069	0.071	0.071	0.071	0.073	0.075	0.076	0.076	0.079	0.079	0.08	0.08	0.084
PSNR	23.87	22.7	22.42	21.12	22.88	22.55	21.99	22.19	22.07	21.29	21.69	20.03	20.62	21.61	21.47	21.37	21.32	21.64	18.65
IoU	0.856	0.843	0.856	0.776	0.791	0.935	0.928	0.92	0.724	0.787	0.498	0.933	0.858	0.86	0.87	0.929	0.599	0.855	0.778

Class	machine	plates	lamp	soundsystem	cup	cabinet	refrigerator	night stand	bathtub	fan	rack	tray	towel	tissues	pen	bowl	oven	desk	phone
MAE	0.085	0.089	0.089	0.09	0.09	0.091	0.092	0.093	0.094	0.097	0.097	0.098	0.098	0.1	0.101	0.106	0.106	0.11	0.114
PSNR	20.44	19.31	20.5	19.62	19.32	21.49	19.17	20.02	19.73	18.78	18.83	18.34	19.23	18.77	18.69	18.39	18.16	18.85	17.73
IoU	0.721	0.953	0.676	0.899	0.819	0.879	0.9	0.794	0.958	0.386	0.651	0.657	0.951	0.769	0.688	0.86	0.914	0.76	0.876

Class	bag	toys	table	blanket	bin	cart	fire extinguisher	faucet	kitchen pan	pillow	stationery	box	utensils	counter	books	tram	electronics	projector
MAE	0.116	0.116	0.117	0.119	0.119	0.12	0.122	0.124	0.128	0.129	0.131	0.138	0.148	0.15	0.153	0.169	0.176	0.185
PSNR	17.58	17.92	18.18	16.69	17.81	16.42	15.06	16.8	16.46	17.1	16.49	16.6	14.99	15.95	15.23	14.06	13.8	12.21
IoU	0.873	0.553	0.682	0.945	0.86	0.852	0.901	0.774	0.867	0.728	0.778	0.842	0.286	0.705	0.773	-	0.883	0.933

Table 8: Quality of NOCS predictions per class: As discussed in Section 5.1 of the paper, we adopt the use of mAE, mPSNR, and mIoU to measure the quality of object NOCS produced by a model on OmniNOCS test set. We provide the split for these numbers per class here. The mIoU for some categories are not available as these do not have ground truth mask annotations in the test set.

E Discussion and limitations

NOCSformer predicts 3D object coordinates aligned to 2D pixel values, yielding 3D-2D correspondences that can be used to estimate the 6DoF object pose. It shows that models that predict 3D-2D correspondences can be scaled to larger datasets and diverse classes, enabling widespread adoptability. This is an alternative to directly regressing the object pose from an image. NOCSformer exhibits both pros and cons compared to methods that directly regress object poses.

More flexible representation: Predicting NOCS allows for different methods to be used for estimating the object pose, based on the application. The options are 1) using learned network heads to predict pose from NOCS (as in NOCSformer) 2) using PnP variants to estimate pose from 3D-2D correspondences 3) using 3D-3D alignment to estimate pose, if a depth sensor is available.

More interpretable: Predicting pose from 3D-2D correspondences is more interpretable than using an end-to-end trained model.

Less accurate at longer ranges: For small objects at very long ranges (such as those in outdoor self-driving scenes), the accuracy of NOCSformer deteriorates. Since the RoI resolution is higher than the size of these objects, the input to the RoI heads is itself coarse and less informative, producing more noisy masks, NOCS and size predictions. The depth estimates at longer ranges are more sensitive to errors in object coordinate predictions, causing higher depth errors. For long ranges, particularly in single-camera applications, it may be more accurate to regress pose directly from an end-to-end model as they rely on dataset biases.

References

1. Ahmadyan, A., Zhang, L., Ablavatski, A., Wei, J., Grundmann, M.: Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021) 2, 5, 6, 19, 24

2. Avetisyan, A., Dahnert, M., Dai, A., Savva, M., Chang, A.X., Nießner, M.: Scan2cad: Learning cad model alignment in rgb-d scans. In: CVPR (2019) 3, 4
3. Avetisyan, A., Dai, A., Nießner, M.: End-to-end cad model retrieval and 9dof alignment in 3d scans. In: ICCV (2019) 3
4. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. CVPR (2022) 6, 7
5. Baruch, G., Chen, Z., Dehghan, A., Dimry, T., Feigin, Y., Fu, P., Gebauer, T., Joffe, B., Kurz, D., Schwartz, A., Shulman, E.: ARKitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile RGB-d data. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1) (2021), https://openreview.net/forum?id=tjZjv_qh_CE 2, 5, 6
6. Birodkar, V., Lu, Z., Li, S., Rathod, V., Huang, J.: The surprising impact of mask-head architecture on novel class segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7015–7025 (2021) 14
7. Brazil, G., Kumar, A., Straub, J., Ravi, N., Johnson, J., Gkioxari, G.: Omni3D: A large benchmark and model for 3D object detection in the wild. In: CVPR. IEEE, Vancouver, Canada (June 2023) 2, 4, 5, 6, 8, 12, 13, 23
8. Brazil, G., Liu, X.: M3d-rpn: Monocular 3d region proposal network for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9287–9296 (2019) 2, 4
9. Cabon, Y., Murray, N., Humenberger, M.: Virtual kitti 2. arXiv preprint arXiv:2001.10773 (2020) 2, 5, 6
10. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020) 2, 6, 9, 19
11. Chen, H., Huang, Y., Tian, W., Gao, Z., Xiong, L.: Monorun: Monocular 3d object detection by reconstruction and uncertainty propagation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021) 2, 4, 9, 23
12. Chen, H., Wang, P., Wang, F., Tian, W., Xiong, L., Li, H.: Epro-pnp: Generalized end-to-end probabilistic perspective-n-points for monocular object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2781–2790 (2022) 2, 12
13. Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals using stereo imagery for accurate object class detection. IEEE transactions on pattern analysis and machine intelligence **40**(5), 1259–1272 (2017) 13
14. Cheng, B., Collins, M.D., Zhu, Y., Liu, T., Huang, T.S., Adam, H., Chen, L.C.: Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In: CVPR (2020) 6
15. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213–3223 (2016) 2, 5, 6, 19
16. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR (2017) 4
17. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020) 2, 7
18. Fu, Y., Wang, X.: Category-level 6d object pose estimation in the wild: A semi-supervised learning approach and a new dataset. In: Advances in Neural Information Processing Systems (2022) 5

19. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012) 2, 5, 6, 19
20. Goodwin, W., Vaze, S., Havoutis, I., Posner, I.: Zero-shot category-level object pose estimation. In: European Conference on Computer Vision. pp. 516–532. Springer (2022) 2, 4, 7
21. Gümeli, C., Dai, A., Nießner, M.: Roca: Robust cad model retrieval and alignment from a single image. In: CVPR (2022) 3
22. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017) 8
23. Heylen, J., De Wolf, M., Dawagne, B., Proesmans, M., Van Gool, L., Abbeloos, W., Abdelkawy, H., Reino, D.O.: Monocinis: Camera independent monocular 3d object detection using instance segmentation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 923–934 (2021) 6
24. Irshad, M.Z., Kollar, T., Laskey, M., Stone, K., Kira, Z.: Centersnap: Single-shot multi-object 3d shape reconstruction and categorical 6d pose and size estimation. In: IEEE International Conference on Robotics and Automation (ICRA) (2022), https://arxiv.org/abs/2203.01929 2
25. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. arXiv:2304.02643 (2023) 6
26. Kundu, A., Genova, K., Yin, X., Fathi, A., Pantofaru, C., Guibas, L., Tagliasacchi, A., Dellaert, F., Funkhouser, T.: Panoptic Neural Fields: A Semantic Object-Aware Neural Scene Representation. In: CVPR (2022) 2, 4
27. Kundu, A., Li, Y., Rehg, J.M.: 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In: CVPR (2018). https://doi.org/10.1109/CVPR.2018.00375, https://doi.org/10.1109/CVPR.2018.00375 2, 8
28. Kuo, W., Angelova, A., Lin, T.Y., Dai, A.: Mask2cad: 3d shape prediction by learning to segment and retrieve. In: ECCV (2020) 3
29. Lepetit, V., Moreno-Noguer, F., Fua, P.: Epnp: An accurate o(n) solution to the pnp problem. International Journal Of Computer Vision **81**, 155–166 (2009). https://doi.org/https://doi.org/10.1007/s11263-008-0152-6, http://infoscience.epfl.ch/record/160138 8
30. Li, K., DeTone, D., Chen, Y.F.S., Vo, M., Reid, I., Rezatofighi, H., Sweeney, C., Straub, J., Newcombe, R.: Odam: Object detection, association, and mapping using posed rgb video. In: ICCV (2021) 3
31. Li, Z., Wang, G., Ji, X.: Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7678–7687 (2019) 2, 8, 9
32. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. CoRR **abs/1405.0312** (2014), http://arxiv.org/abs/1405.0312 19
33. Liu, Z., Zhou, D., Lu, F., Fang, J., Zhang, L.: Autoshape: Real-time shape-aware monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15641–15650 (2021) 2, 4
34. Manhardt, F., Kehl, W., Gaidon, A.: Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2069–2078 (2019) 3
35. Maninis, K.K., Popov, S., Nießner, M., Ferrari, V.: Vid2cad: Cad model alignment using multi-view constraints from videos. TPAMI (2022) 3
36. Maninis, K.K., Popov, S., Nießner, M., Ferrari, V.: Cad-estate: Large-scale cad model annotation in rgb videos. In: ICCV (2023) 4

37. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020) 6
38. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 7074–7082 (2017) 3
39. Nie, Y., Han, X., Guo, S., Zheng, Y., Chang, J., Zhang, J.J.: Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 55–64 (2020) 2, 4
40. Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023) 2, 7, 23
41. Rad, M., Lepetit, V.: Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In: Proceedings of the IEEE international conference on computer vision. pp. 3828–3836 (2017) 2, 3
42. Rajasegaran, J., Pavlakos, G., Kanazawa, A., Malik, J.: Tracking people by predicting 3D appearance, location & pose. In: CVPR (2022) 2
43. Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 12179–12188 (2021) 7, 23
44. Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M.A., Paczan, N., Webb, R., Susskind, J.M.: Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In: International Conference on Computer Vision (ICCV) 2021 (2021) 2, 5, 6, 19
45. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 567–576 (2015). https://doi.org/10.1109/CVPR.2015.7298655 2, 5, 6, 19
46. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020) 2, 5, 6, 19
47. Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., Xue, T., Tenenbaum, J.B., Freeman, W.T.: Pix3d: Dataset and methods for single-image 3d shape modeling. In: CVPR (2018) 4
48. Wang, G., Manhardt, F., Tombari, F., Ji, X.: Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16611–16621 (2021) 2, 4, 8, 9
49. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019) 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14
50. Wang, T., Xinge, Z., Pang, J., Lin, D.: Probabilistic and geometric depth: Detecting objects in perspective. In: Conference on Robot Learning. pp. 1475–1485. PMLR (2022) 12
51. Wang, T., Zhu, X., Pang, J., Lin, D.: Fcos3d: Fully convolutional one-stage monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 913–922 (2021) 2, 12
52. Wang, Y., Guizilini, V.C., Zhang, T., Wang, Y., Zhao, H., Solomon, J.: Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In: Conference on Robot Learning. pp. 180–191. PMLR (2022) 3
53. Wen, B., Lian, W., Bekris, K., Schaal, S.: Catgrasp: Learning category-level task-relevant grasping in clutter from simulation. ICRA 2022 (2022) 4

54. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. arXiv preprint arXiv:1711.00199 (2017) 2, 3
55. Xu, G., Wang, X., Ding, X., Yang, X.: Iterative geometry encoding volume for stereo matching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21919–21928 (2023) 6
56. Zhang, J., Herrmann, C., Hur, J., Cabrera, L.P., Jampani, V., Sun, D., Yang, M.H.: A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. arXiv preprint arxiv:2305.15347 (2023) 7
57. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5745–5753 (2019) 8