RayTran: 3D pose estimation and shape reconstruction of multiple objects from videos with ray-traced transformers

M. J. Tyszkiewicz[2⋆], K.-K. Maninis[1], S. Popov[1], and V. Ferrari[1]

[1]Google Research [2]EPFL

Abstract. We propose a transformer-based neural network architecture for multi-object 3D reconstruction from RGB videos. It relies on two alternative ways to represent its knowledge: as a global 3D grid of features and an array of view-specific 2D grids. We progressively exchange information between the two with a dedicated bidirectional attention mechanism. We exploit knowledge about the image formation process to significantly sparsify the attention weight matrix, making our architecture feasible on current hardware, both in terms of memory and computation. We attach a DETR-style head [9] on top of the 3D feature grid in order to detect the objects in the scene and to predict their 3D pose and 3D shape. Compared to previous methods, our architecture is single stage, end-to-end trainable, and it can reason holistically about a scene from multiple video frames without needing a brittle tracking step. We evaluate our method on the challenging Scan2CAD dataset [3], where we outperform (1) state-of-the-art methods [39,34,35,15] for 3D object pose estimation from RGB videos; and (2) a strong alternative method combining Multi-View Stereo [17] with RGB-D CAD alignment [4].

1 Introduction

Detecting and reconstructing objects in 3D is a challenging task with multiple applications in computer vision, robotics, and AR/VR that require semantic 3D understanding of the world. In this paper we propose RayTran, a transformer-based [59] neural network architecture for reconstructing multiple objects in 3D given an RGB video as input. Our key new element is a backbone which infers a global representation of the 3D volume of the scene. We attach a DETR-style head [9] on top of it, which detects objects in the 3D representation and predicts their 3D pose and shape (Figure 1).

The backbone inputs multiple video frames showing different views of the same static scene. Its task is to jointly analyze all views and to consolidate the extracted information into a global 3D representation. Internally, the backbone maintains two alternative scene representations. The first is three-dimensional and describes the volume of the scene. The second is two-dimensional and describes the volume from the perspective of the individual views. We connect

[⋆] Work done while at Google Research.

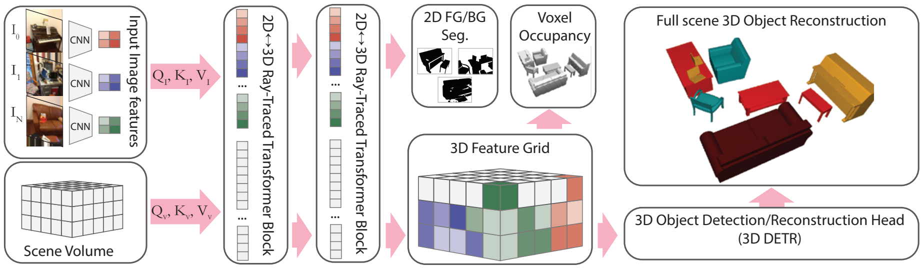
I0
I1
IN
CNN
CNN
CNN
Input Image features
Scene Volume
QI, KI, VI
QV, KV, VV
2D↔3D Ray-Traced Transformer Block
2D↔3D Ray-Traced Transformer Block
2D FG/BG Seg.
Voxel Occupancy
3D Feature Grid
Full scene 3D Object Reconstruction
3D Object Detection/Reconstruction Head (3D DETR)

Fig. 1: **Overview of our method:** The RayTran backbone processes information in two parallel network streams. The first one (2D) works on features extracted on the multiple input frames. The second one (3D) starts from an empty volumetric feature representation of the scene. The 2D stream gradually consolidates features on the 3D volume and visa-versa with repeated blocks of ray-traced transformers. The backbone outputs a 3D feature grid which offers a global representation of the 3D volume of the scene. We attach a DETR-style head [9] to this representation, to detect all objects in 3D and to predict their 3D pose and 3D shape. We further help training with two auxiliary tasks: predicting 3D coarse binary occupancy for all objects together, and predicting amodal 2D foreground-background masks.

these two representations with a bidirectional attention mechanism to exchange information between them, allowing the 3D representation to progressively accumulate view-specific features, while at the same time the 2D representation accumulates global 3D features.

Processing videos with transformers is notoriously resource-consuming [7,2]. Our case is no exception: if we relied on attention between all elements in the 2D and 3D representations, the attention matrix would have infeasible memory requirements (and it would also be computationally very expensive). To overcome this, we propose a *sparse ray-traced attention* mechanism. Given the camera parameters for each view, we exploit the image formation process to identify pairs of 2D and 3D elements that are unlikely to interact. We omit these pairs and store the attention matrix in a sparse format. This greatly reduces its computational and memory complexity, by a factor of $O(|V|^{\frac{2}{3}})$, where $|V|$ is the number of voxels in the 3D representation.

We attach a DETR-style head [9] on top of the 3D representation produced by the backbone. This head detects objects and predicts their class, 3D shape, and 3D pose (translation, rotation, scale). We represent object shapes with a voxel grid and then extract meshes using marching cubes [33]. We also predict coarse binary volumetric occupancy for all objects together, using a 3D convolutional layer on top of the global 3D representation. This provides an auxiliary task that teaches the network about the scene's geometry, and is essential for training.

As a second auxiliary task, we add an additional network head that predicts the 2D amodal foreground-background binary masks of all objects in the scene. Besides enabling this task, this head also helps training the backbone as it closes the loop between images and the 3D representation.

Several recent works [39,52,34] tackle 3D scene reconstruction from videos in the same setting. They rely on a 3-step pipeline: (1) object detection in individual

2D frames, along with estimating properties such as 3D rotation, parts of 3D scale, and 3D shape (either as a parametric surface [34] or by retrieving a CAD model from a database [39]); (2) tracking-by-detection [1,8,6], to associate 2D detections across frames; (3) multi-view optimization to integrate the per-frame predictions. This completes all 3D pose parameters, resolving the scale-depth ambiguities, and places all objects in a common, global 3D coordinate frame.

Our method was inspired by these works and addresses several of their shortcomings. The pipelines are composed of heterogeneous steps, which are trained separately and require manual tuning to work well together. The pipelines are complicated and over-engineered due to the intricate nature of the full-scene object reconstruction task. The tracking step is especially brittle. Objects often go out of view and re-appear later, and occlude each other over time. This poses a major challenge and leads to objects broken into multiple tracks, as well as tracks mixing multiple objects. These tracking errors harm the quality of the final 3D reconstructions.

In contrast, our method is end-to-end trainable. It is built from well understood neural network modules and it has a simple, modular architecture in comparison. Importantly, *we avoid tracking altogether*. Furthermore, our method does not rely on any notion of time sequence, so it is also applicable to sparse multi-view inputs (in addition to video).

We evaluate RayTran on the challenging Scan2CAD [3] dataset, featuring videos of complex indoor scenes with multiple objects. Through extensive comparisons we show that RayTran outperforms several works: (1) two baselines that process frames individually, defined in [39] as extensions of Mask2CAD [31]. This illustrates the value of jointly processing multiple frames in RayTran; (2) four recent multi-frame methods Vid2CAD [39], ODAM [34], MOLTR [35], ImVoxNet [15]. Besides performing better, RayTran also offers a much simpler design than [39,34,35], with an end-to-end trainable, unified architecture which does not require a tracking module; (3) a strong alternative method that combines the state-of-the-art Multi-View Stereo [17] and RGB-D CAD alignment [4] methods.

2 Related Work

3D from multiple views. Classic SfM/SLAM works cast 3D reconstruction as estimation of 3D points from multiple views based on keypoint correspondences [49,43,61,54,18]. However, the output point cloud is not organized into objects instances with their classes, 3D shapes, or poses. A line of works detect and localize objects in 3D using multi-view projection constraints, by approximating the object shapes with 3D boxes [64] and ellipsoids [45]. ODAM [34] goes a step further to creates a scene representation out of superquadrics, by using a graph neural network as core architecture for object association in time. FroDO [52] and MO-LTR [35] rely on both 2D image cues and the sparse 3D point clouds from SfM/SLAM to reconstruct objects in the scene. Qian et al. [51] produce volumetric reconstructions of multiple objects in a synthetically generated scene. Vid2CAD [39] integrates the single-view predictions of Mask2CAD [31] across time, to place objects from a CAD database into the 3D scene.

A common caveat of multi-view methods for 3D object reconstruction is that their architectures are overly complex, they cannot be trained end-to-end due to their heterogeneity, and they often rely on a brittle tracking-by-detection step. Instead, our proposed method provides a light-weight end-to-end architecture for the task, while we completely avoid tracking.

Similar to RayTran, the concurrent ImVoxelNet [15] keeps its 3D knowledge in a global 3D representation and does not require tracking. It uses a hand-crafted unidirectional mechanism to project and consolidate image features onto it. In contrast, our ray-traced transformers learn the optimal way to consolidate features. They are also bidirectional, which enables 2D supervision through re-projection as well as additional tasks, like novel-view synthesis. Moreover, RayTran reconstructs the 3D shapes of the detected objects, going beyond detecting 3D boxes.

Transformer architectures for computer vision. Several recent works use attention-based architectures (transformers) [59] for computer vision tasks. ViT [16] replaces the traditional convolutional backbones with attention among patches for image classification. The same idea has been incorporated into network designs for semantic segmentation [57,65,12], object detection [9], and panoptic segmentation [12]. Transformers have been introduced recently also for video processing. TrackFormer [40] uses a transformer architecture for multi-object tracking. ViViT [2] and TimeSFormer [7] use ViT-like patches from multiple frames for video classification.

The main bottleneck of these approaches are the prohibitive memory requirements. TrackFormer [40] can only process 2 images at a time, which prevents end-to-end training on the whole video. Similarly, the all-to-all patch attention, which is the cornerstone of [7,2], comes with often infeasible memory requirements. ViViT [2] needs the combined memory of 32 TPU accelerators to process a single batch of 128 frames. Our work overcomes these limitations by using sparse attention between 2D and 3D features. The sparsity is achieved by using image formation constraints directly from the poses of the cameras, which significantly reduces the memory requirements. For reference, RayTran processes up to 96 frames of a video and reconstructs all instances on a single 16 GB GPU.

3D using a dedicated depth sensor. Our work draws inspiration from several 3D object reconstruction methods that directly work on point clouds obtained by fusing RGB-D video frames. Early works use known pre-scanned objects [53], hand-crafted features [44,21,36,56], and human intervention [56]. Recent works use deep networks to directly align shapes on the dense point clouds [3,4,5,28,55]. Fei et al. [20] align a known set of shapes on a video in 4 DoF, by using a camera with an inertial sensor.

Using an additional sensor reduces the search-space required to accurately re-construct an object in 3D. Both the depth and the inertial sensors eliminate the depth-scale ambiguity, and compared to re-constructing from pure RGB, RGB-D sensors provide cleaner, much more realistic results. Our work does not

require the intermediate step of point-based reconstruction, does not use the extra depth sensor, and can directly reconstruct objects in a posed RGB video.
3D detection and reconstruction from a single image. Pioneering works in this area process a single image to either infer the pose of an object as an oriented 3D bounding box [42,38], or to also predict the 3D shape of the object [60,41,13,22,62,63,47,11]. Works that are able to predict an output for multiple object instances, typically first detect them in the 2D image, and then reconstruct their 3D pose and/or shape [27,23,31,29,58,30,46,50,32,19,26,24].

3D predictions from single images tend to be inaccurate due to scale-depth ambiguity, and often methods of this category compensate for it in a variety of ways, e.g., based on estimating an approximate pixel-wise depth map from the input image [27], by requiring manually provided objects' depth and/or scale [23,31,32] at test time, or by estimating the position of a planar floor in the scene and assuming that all objects rest on it [29]. Some works [58,46,50] attempt to predict object depth and scale directly based on image appearance. Our proposed approach processes multiple frames simultaneously, and implicitly compensates for the scale-depth ambiguity by using many different view-points of the objects appearing in the scene.

3 Proposed Approach

Our method takes multiple views (video frames) of a scene and their camera parameters as input. Each view captures a different part of the same 3D scene. It outputs the 3D pose (rotation, translation, scale), the class, and the 3D shape of all objects in the scene.

We achieve this with a single, end-to-end trainable, neural network model. We propose a transformer-based backbone that processes the input views and infers a global 3D volume representation for the entire scene. We use this representation to predict the object shapes, poses, and classes, by attaching a DETR-style [9] head to it. In addition, we perform two additional auxiliary tasks: 3D occupancy, where we predict coarse binary volumetric occupancy for all objects together, and 2D foreground-background amodal segmentation. The overview of our architecture is illustrated in Fig. 1.

3.1 The RayTran Backbone

We propose a neural network architecture that operates on two alternative representations in parallel. The first one is three-dimensional and describes the 3D space that the scene occupies. We use a voxel grid V with *global* features that coincides with this space. The second one is two-dimensional and describes the scene from the perspective of the individual views. For each view $i = 1..N$, we use a pixel grid P_i of *image* features that coincides with the view's image.

The two representations are connected implicitly through the image formation process. We model this as a sequence of $2D \Leftrightarrow 3D$ neural network transformer blocks (Figure 2, left). The j-th block takes all views $P^j_{1...N}$ and the volume V^j as input, mixes their features, and outputs a pair of new representations

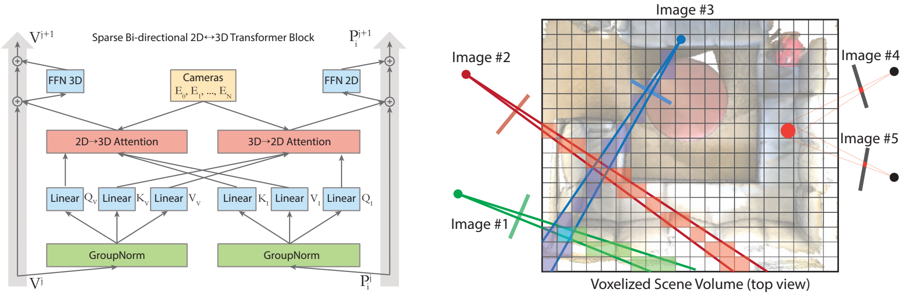

Fig. 2: **$2D$ ⇔ $3D$ ray-traced transformer block (left).** Each block uses two parallel residual network streams that exchange information by attention. They consist of two layers of ray-traced sparse attention (2D→3D and 3D→2D) followed by a feed-forward network (FFN) composed of 3D and 2D convolutions, respectively. The voxel features (3D) inform the image features (2D) at each stage of the backbone. The 3D reconstruction head uses the voxel features (output of left stream), whereas the 2D foreground-background segmentation head uses the pixel grid (output of right stream). **Intertwining 3D voxel- with 2D image features (right):** Multiple voxels can project on the same pixel, and multiple pixels from multiple cameras can look at the same voxel. The proposed attention layer models this interaction in an intuitive way.

($P_{1...N}^{j+1}$ and V^{j+1}). This allows the global 3D representations to be progressively populated by local features from the different views, while at same time the 2D representations progressively accumulate global features in different depths of the network.

The output of the RayTran backbone is a 3D feature representation of the scene, derived from the input views. In order to compute the initial 2D representation P_i^0, we embed ResNet-18 [25] in our backbone (pre-trained on ImageNet). We run ResNet-18 over the input views i and we take the output of its penultimate block for each view. To initialize the 3D volume representation V^0, we cast a ray (un-project) from all the pixels P_i^0 onto the 3D volume. We then average the image features that fall into each voxel of V^0.

Block operation. The 2D⇔3D blocks of RayTran consist of two parallel network streams, as shown in Figure 2 (left). The first one ($2D \Rightarrow 3D$), mixes features from P_i^j into V^j and outputs V^{j+1}. The second one ($3D \Rightarrow 2D$) from V^j into P_i^j, resulting in P_i^{j+1}. We propose to build both networks using the multi-headed attention mechanism [59].

The attention mechanism can translate an input vector (1D array of features) from a source domain into a differently-sized vector in a target domain. To do this, the mechanism computes a *key* vector that describes each position in the source domain and a *query* vector that describes each position in the target domain. It then computes a matrix describing the relation between source and target positions, by storing the dot product between the features at position i in the *key* and position j in the *query* at (i, j) in the matrix. Finally, the mechanism computes a *value* vector from the input vector and multiplies this with the attention matrix in order to obtain the output. The *key* and the *value* depend on the input vector (from the source domain), while the *query* depends on a vector from the target domain. The goal of the attention mechanism is to learn the dependencies between the two domains.

The attention mechanism is intrinsically well suited to model the connection between pixels and voxels. Multiple pixels from multiple cameras can look at the same voxel, as shown in Fig. 2 (right). We need a mechanism to consolidate their features in the voxel. Similarly, multiple voxels can project onto the same pixel and we need to consolidate their features. The matrix-*value* multiplication in the attention mechanism naturally achieves the desired effect.

For $2D \Rightarrow 3D$ attention, we derive the *key* and the *value* from all pixels from all views of P_i^j and the *query* from all voxels of V^j. For $3D \Rightarrow 2D$ attention, conversely, from V^j and P_i^j. We introduce skip connections in both networks, by adding the inputs of the attention mechanism to its outputs. We then post-process with a feed-forward network, built with 3D and 2D convolution layers respectively (Fig. 2).

Ray-traced attention layers. In a realistic setting, the attention layer has infeasible memory requirements. Our backbone operates on multiple frames simultaneously, 20 during training and 96 at inference time, each using a 2D feature grid of 40×30 for the 2D features P_i. We use a voxel grid with resolution $48 \times 48 \times 16$ to model a $9m \times 9m \times 3.5m$ volume, corresponding to voxel dimensions of approximately $19cm \times 19cm \times 22cm$. We use 256 features in both the 2d and 3d representations and 8 heads in the attention layers. Given the above numbers, the attention matrices in each 2D $\Leftrightarrow$ 3D block alone would require $\approx$ 52GB of memory with 20 frames, which is prohibitive.

To overcome this, we embed knowledge about the image formation process into the architecture (Fig. 3). A pixel and a voxel can interact with each other directly only if there is a camera ray that passes through both of them. If no such ray exists, the two are unlikely to interact, and we set the corresponding entry in the attention matrix to zero. This is mathematically equivalent to the masking mechanism employed in autoregressive transformers to enforce causality [59], but crucially allows us to store the matrix in sparse form and significantly reduce memory consumption. A pixel can only interact with $O(\sqrt[3]{|V|})$ voxels, where $|V|$ is the number of voxels in V, since any ray can only pass through at most this many voxels. We thus need $O(|V|^{\frac{2}{3}})$ times less memory to store the matrix. In sparse coordinate format, which encodes each matrix entry with 3 numbers (row, column, value), the matrix from our example above would consume 270 times less memory (3×64.4MB instead of 52GB). We call multi-headed attention based on such sparse matrices *ray-traced sparse attention.*

We use the camera parameters to determine which pixel-voxel pairs interact with each other. In turn, the camera parameters can be computed with off-the-shelf pipelines such as COLMAP [54]. To make full use of the limited volume that our backbone can focus on, we center the camera positions within it.

3.2 Task-specific heads on top of the backbone

3D pose estimation and shape reconstruction. For our main task, we predict the 3D pose, and reconstruct the shape of all objects seen in the video.

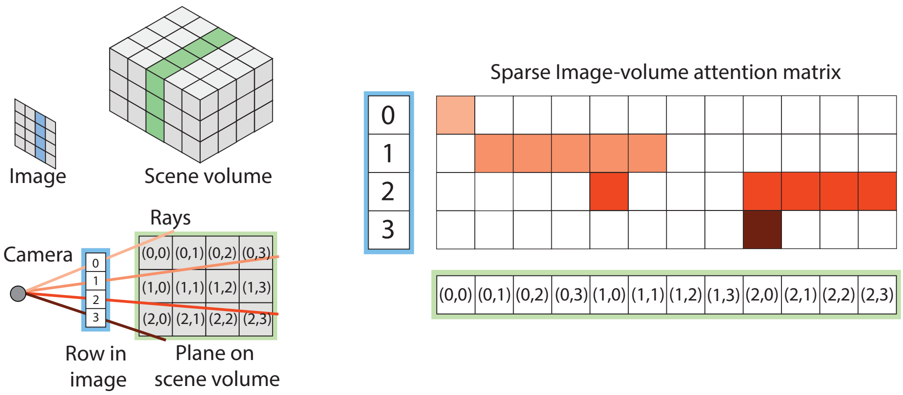

Fig. 3: **Ray-traced sparse attention.** A pixel and a voxel are likely to interact only if a ray passes through both of them. We exploit this to significantly reduce the memory requirements of our 2D ⇔ 3D blocks, by sparsifying the attention matrices. If no ray passes though a pixel/voxel pair, the two are unlikely to interact and we omit the corresponding value in the matrix.

We use a DETR-style [9] architecture, with multi-headed attention between 64 object query slots and the voxels in the backbone output. In each slot, we predict the object's class, its shape in canonical pose, 3D center, 3D anisotropic scale, and 3D rotation. We use a special *padding* class to indicate that a query slot does not contain a valid object. We encode the shape as a 63×63×63 voxel grid, which we predict with a sequence of transposed 3D convolution layers from the query's embedding. We use Marching Cubes [33] to convert the voxel grid to a mesh. We only predict rotation around the 'up'-axis of each object (as one angle), as most objects in our dataset are only rotated along this axis.

We use cross entropy for predicting the class, binary cross entropy for the shape's voxels, soft L_1 loss for the object center, L_1 loss over the logarithm of the scales, and a soft L_1 loss for the rotation angles. Finally, we match predictions to ground-truth objects in DETR using a linear combination of all losses except the voxel one, which we exclude for performance reasons. As in [9], we supervise at all intermediate layers.

3D occupancy prediction. As an auxiliary task, we also predict the binary 3D occupancy of all objects for the whole scene on a coarse voxel grid. We use one 3D convolution layer on top of the backbone output, and we supervise with binary-cross-entropy. We run occupancy prediction as an auxiliary task, to directly teach the network about the combined object geometry. This is crucial for the 3D object reconstruction task, as the DETR-style head fails to pick up any training signal if the network is trained without it.

2D foreground-background (FG/BG) segmentation. As a second auxiliary task, we predict a 2D *amodal* segmentation mask in each view, for all objects together. A pixel belongs to the mask if it lies on any object in the view, regardless of occlusion. We use transposed convolutions, combined with non-linearities and normalization layers, to up-sample the pixel stream output P_i^n of the last 2D ⇔ 3D block to the original input resolution (16-fold). We supervise

Family	Method	**class avg.**	**global avg.**	bathtub	bookshelf	cabinet	chair	display	sofa	table	trashbin	other
Single-frame baselines	Mask2CAD [31] +avg	2.5	3.5	0.0	1.9	1.5	6.8	3.7	2.7	1.4	3.0	1.2
	Mask2CAD [31] +pred	11.6	16.0	8.3	3.8	5.4	30.9	17.3	5.3	7.1	25.9	0.5
Multi-Frame Methods	MVS [17] + RGB-D fitter [4]	18.8	21.7	15.8	8.5	17.3	34.3	25.7	15.0	10.9	35.8	6.1
	ODAM [34]	25.6	29.2	24.2	12.3	13.1.	42.8	36.6	28.3	31.1	42.2	0.0
	Vid2CAD [39]	30.7	38.6	28.3	12.3	23.8	64.6	37.7	26.5	28.9	47.8	6.6
	RayTran	**36.2**	**43.0**	19.2	34.4	36.2	59.3	30.4	44.2	42.5	31.5	27.8

Table 1: Quantitative results on the Scan2CAD [3] dataset using the original Scan2CAD metrics. Results for Mask2CAD variants, MVS+RGB-D fitter, and Vid2CAD are as reported in [39]. ODAM originally reports in another metric (Tab. 2). We re-evaluate in the Scan2CAD metrics based on model outputs provided to us by the authors. Note that ODAM was not trained to predict the 'other' class. When excluding it from the metrics, ODAM achieves class avg. of 28.8% and global avg. of 33.5%.

using the binary cross entropy loss. We create the amodal masks by rasterizing the combined geometry of all ground-truth 3D objects into each view. Predicting amodal masks enhances the backbone's 3D understanding of the world. In general, the amodal mask is ill-defined for occluded regions in a single image. It becomes well defined with multiple views however, if some of them observe the object behind the occluder. Hence, this FG/BG task pushes our network to reason about geometric relations across multiple views.

Novel View Synthesis. While we focus on multi-object 3D reconstruction, our backbone and the scene-level representation it outputs can be used for other 3D tasks as well. In the supplemental material, we provide qualitative results for Novel View Synthesis, which builds upon RayTran's backbone.

4 Experiments

Datasets and evaluation metrics. We evaluate our method on Scan2CAD [3], following their protocol and evaluation metrics. Concretely, we use videos from ScanNet [14], 3D CAD models from ShapeNetCore [10], and annotations that connect the two from Scan2CAD [3]. ScanNet provides videos of rich indoor scenes with multiple objects in complex spatial arrangements. ShapeNetCore provides CAD models from 55 object classes, in a canonical orientation within a class. Scan2CAD provides manual 9-DoF alignments of ShapeNetCore models onto ScanNet scenes for 9 super-classes.

We use these datasets both for training and evaluation. During training, we consider all ScanNet videos in the official train split whose scenes have Scan2CAD annotations (1194 videos). We evaluate on the 306 videos of the validation set, containing a total of 3184 aligned 3D objects. We quantify performance using the original Scan2CAD metrics [3] and the metrics introduced in ODAM [34]. In the Scan2CAD metrics, a ground-truth 3D object is considered accurately detected if one of the objects output by the model matches its class and pose alignment (passing three error thresholds at the same time: 20% scale,

Prec./Rec./F1	MOLTR [35]	ODAM [34]	Vid2CAD [39]	ImVoxelNet [15]	RayTran
@IoU> 0.25	54.2/55.8/55.0	64.7/58.6/61.5	56.9/55.7/56.3	52.9/53.2/53.0	**65.4**/**61.8**/**63.6**
@IoU> 0.5	15.2/17.1/16.0	31.2/28.3/29.7	34.2/33.5/33.9	17.0/17.1/17.0	**41.9**/**39.6**/**40.7**

Table 2: Quantitative results on Scan2CAD using the ODAM [34] metrics. To evaluate RayTran, we derive an oriented 3D box by using the 3D transformations predicted by the model. RayTran outperforms all other works, especially at the stricter IoU threshold (@IoU> 0.5), showing it produces particularly accurate object poses. Note that for Vid2CAD we report the updated results from `https://github.com/likojack/ODAM` (which match exactly the Vid2CAD paper [39]). Also note that ImVoxelNet outputs axis-aligned boxes, which hinders its performance at high IoU thresholds. Finally, results for MOLTR and ODAM are as reported in [34].

20° rotation, 20cm translation). We report accuracy averaged over classes ('class avg.') as well as over all object instances ('global avg'). In the metrics of [34], an object is considered accurately detected if the Intersection-over-Union (IoU) of its oriented 3D bounding box to a ground-truth box of the same class is above a predefined threshold. We report precision, recall, and F1 score. Finally, the dataset also provides dense 3D meshes for the scene produced using a dedicated depth sensor. We ignore this data, both at training and test time (in contrast to some previous works which rely on it [3,4,5,28,55]).

Training details. We implement our model in PyTorch [48]. We train on 20 frames per video, using 16-bit float arithmetic. This allows us to fit one video on a GPU with 16GB of memory. We use 8 GPUs in total, resulting in a batch size of 8. We train RayTran in three stages. We first train just the backbone for 224k steps (1500 epochs) on the task of predicting 3D occupancy (Sec. 3.2). We then enable all other tasks except the shape predictor and we train for another 239k steps (1604 epochs). Finally, we train just the shape predictor for another 5k steps (17 epochs), after freezing the rest of the network parameters. We use the AdamW [37] optimizer, with a learning rate of 10^{-4} and weight decay $5 \cdot 10^{-2}$.

Compared methods. We compare RayTran against Vid2CAD [39], ODAM [34], MOLTR [35], and ImVoxelNet [15], four recent methods for 3D object pose estimation and detection from RGB videos.

We further compare to two baselines that process frames individually, defined by [39]. These extend Mask2CAD [31], which in its original form does not predict the 3D depth nor the scale of the object. The first baseline, 'Mask2CAD +avg', estimates an object's depth and scale by taking the average over its class instances in the training set. The second baseline, 'Mask2CAD +pred', predicts the scale of the actual object in the image (and then derives its depth from it). Both baselines aggregate 3D object predictions across all video frames and remove duplicates that occupy the same volume in 3D.

Several previous methods report strong results on Scan2CAD by using a dedicated RGB-D depth sensor to acquire a dense 3D point-cloud of the scene. Those methods have an intrinsic advantage and operate by directly fitting CAD models on the scene's 3D point cloud [3,4,5,28]. Instead, our method only uses the RGB frames. Hence, we compare to a strong alternative method, defined in [39], that replaces the input of the best RGB-D fitting method [4] with 3D point-

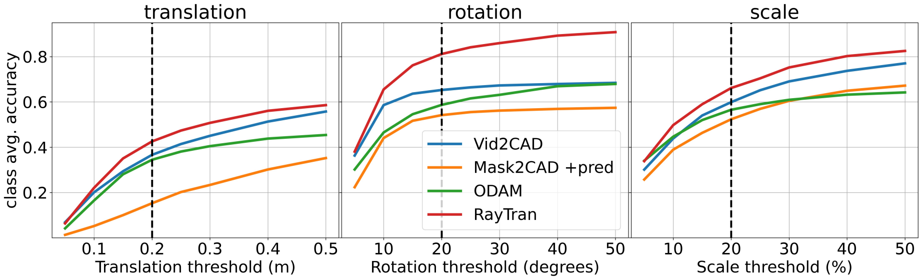

Fig. 4: **Transformation type ablation**: Class-avg accuracy as a function of the evaluation threshold (the vertical dotted line shows the default value, used in Tab. 1). We examine each transformation type separately. RayTran achieves better accuracy than Vid2CAD, ODAM, and 'Mask2CAD +pred' on all transformation types.

clouds generated by the state-of-the-art multi-view stereo method DVMVS [17]. We train DVMVS on ScanNet, and re-train [4] on its output.

Main results. Tab. 1 shows the results in the Scan2CAD metrics. RayTran outperforms both single-frame baselines as well as the 'MVS + RGBD fitter' combination by a wide margin (+33.7%, +24.6%, +17.4% class avg. accuracy respectively). RayTran also outperforms both competitors that align CAD model to RGB videos but rely on tracking: Vid2CAD [39] (+5.5%) and ODAM [34] (+10.6%). Importantly, RayTran is also much simper in design, as [39,34] consist of multiple disjoint steps (object detection, tracking, multi-view optimization). Fig. 5 and Fig. 6 illustrate qualitative results for our method.

Looking at individual categories, we obtain the best result on 5 out of 9, and in particular on the "other" category, which is hard for methods based on retrieving CAD models [39,3,4]. Our method instead predicts 3D shapes as voxel grids which helps to generalize better and to adapt to the large variety of object shapes in this catch-all category. On 'trashbin' we do moderately worse, possibly because of the relatively coarse voxel resolution of the backbone representation.

For completeness, we also compare to methods [3,4] in their original form, i.e. fitting CAD models to high-quality dense RGB-D scans. Surprisingly, RayTran (36.2%/43.0%) improves over [3] (35.6%/31.7%), despite using only RGB video as input. While the state-of-the-art [4] performs even better (44.6%/50.7%), this family of methods are limited to videos acquired by RGB-D sensors.

Fig. 4 reports accuracy for each transformation type separately (translation, rotation, and scales). Our method predicts all transformation types better than Vid2CAD, ODAM, and the best single-frame baseline Mask2CAD+pred. As objects are considered accurately detected only when passing all 3 thresholds *simultaneously*, improving translation is the biggest avenue for improving our overall quantitative results (Tab. 1).

Tab. 2 reports results in the ODAM metrics, which allow us to compare to MOLTR [35] and ImVoxelNet [15]. We choose an object score threshold to maximize the F1 score on the val set for methods that predict object scores (RayTran, Vid2CAD, ImVoxelNet), following the practice of [34]. RayTran outperforms all four methods [34,35,15,39] at both IoU thresholds. ImVoxelNet [15]

Method.	extra input	auxiliary tasks	class avg.	global avg.
RayTran	-	3D occupancy + 2D FG/BG seg.	36.2	43.0
RayTran w/o FG/BG.	-	3D occupancy	33.8	40.1
RayTran + GT masks	2D GT masks	3D occupancy + 2D FG/BG seg.	47.6	52.5

Table 3: Effects of object segmentation in RayTran. Without FG/BG segmentation as an auxiliary task, the network performs worse (first two rows). If we grant the model the ground-truth segmentation masks as input, results substantially improve, highlighting how future progress on automatic 2D segmentation will benefit our work too (last row). In all cases, the model is trained with the main 3D object pose/shape estimation loss (Sec. 3.2), in addition to the auxiliary losses listed here.

reports results on ScanNet, not on Scan2CAD. To compare properly we use their publicly available source code and re-train on Scan2CAD. The original code only outputs axis-aligned object boxes on ScanNet (and hence on Scan2CAD, which is derived from it). This prevents comparison on the Scan2CAD metrics, as we cannot compute precise rotation and scale components. Finally, predicting box rotation could potentially improve the results in the ODAM metrics.

Ablation: 2D FG/BG segmentation as auxiliary task. Our model predicts amodal masks, which reinforces the backbone's 3D understanding. Pixels where the object is occluded can only be predicted correctly in 2D as part of the amodal mask by relying on signal from other frames, via the global 3D representation. To support this claim, we trained a version of our model where we disabled the 2D FG/BG segmentation auxiliary task of Sec. 3.2. This reduces class-avg accuracy by -2.4% (36.2% vs. 33.8 first two rows of Tab. 3).

Ablation: Perfect segmentation. Our model performs both 2D and 3D analysis. The main challenge in the 2D analysis is pixelwise segmentation in the input frames. We explore here what would happen if our model were granted perfect object segmentation as input. We train a model which inputs a binary mask as a 4th channel, in addition to RGB. A pixel in the mask is on if it belongs to any object of the 9 classes annotated in Scan2CAD, and 0 otherwise. This augmented model improves class-avg accuracy by +11.4% (reaching 47.6%), and global-avg by +9.5% (reaching 52.5%, Tab. 3 last row). Hence, as research on 2D segmentation improves, so will our model's 3D scene understanding ability.

Ablation: Number of objects in the scene. By design, our network cannot predict more object instances than the query slots in the DETR head (64). Moreover, typically only about 30% of all query slots bind to an actual object [9] in scenes containing many objects. This limits recall on such scenes, which sometimes do occur in Scan2CAD. Carion et al [9] believe that query slots tend to bind to fixed spatial regions, regardless of the content of a test image, causing this limitation. We operate in 3D, which likely exacerbates it because we need many more queries to cover the 3D space.

To understand the effect of this phenomenon on our model's performance, we evaluate here on 3 subsets of Scan2CAD's val split, containing scenes with *at most* 10, 20, and 30 objects respectively. This reduces the number of objects undetected by the fixed 64 query slots in our DETR head.

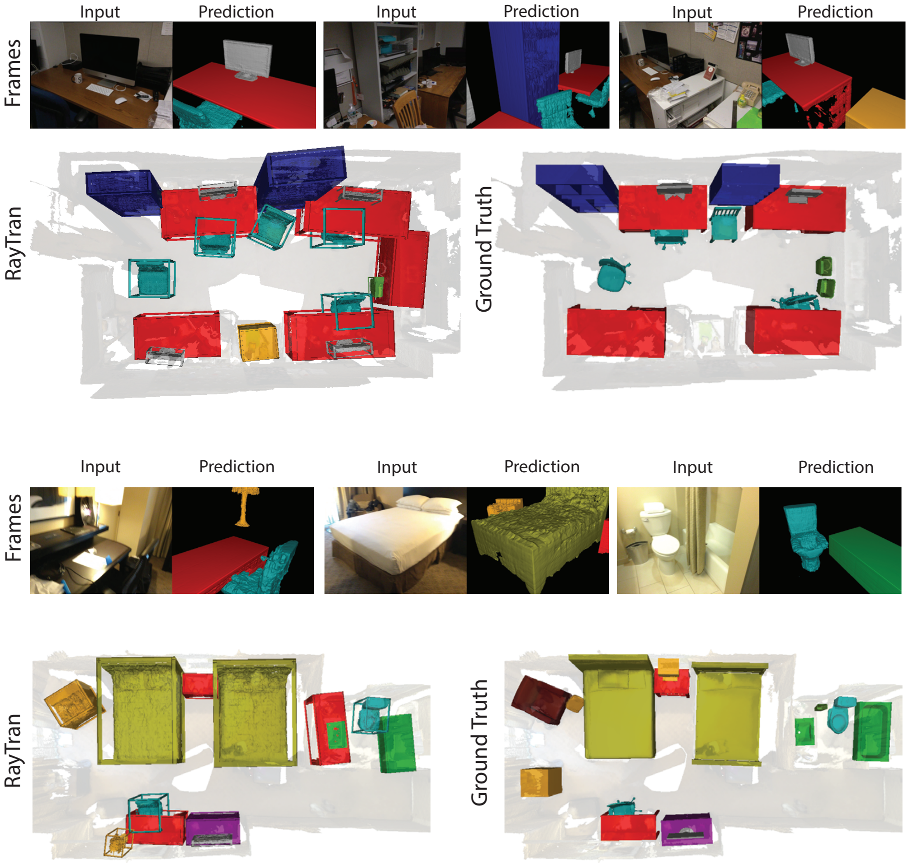

Fig. 5: **Qualitative Results (top-view, with frame overlays)**: We show the 3D pose estimation (as oriented boxes) and shape reconstruction outputs of RayTran against the ground truth, from the top and from the viewpoint of the images. The objects are colored by class. We are able to reconstruct complex scenes in a single pass.

The class-avg accuracy of RayTran indeed improves in scenes containing fewer objects (from 36.2% in all scenes, up to 37.4% in scenes with $<$ 10 objects). The accuracy of the best previous method Vid2CAD instead remains constant. In scenes with at most 10 objects, we outperform Vid2CAD by 6.7% (37.4% vs. 30.7%), which is a larger difference than on all scenes (5.5%: 36.2% vs. 30.7%). Hence, DETR's limitation is affecting our model as well and improving upon it will improve our overall performance.

Ablation: Number of input frames. Our model can process a variable number of input frames per video. We use 20 frames at training time to limit memory requirements. In all experiments so far, we used 96 frames at inference time, as using more frames improves coverage of the 3D volume of the scene and hence accuracy of the output. To support this claim, we now reduce the number of

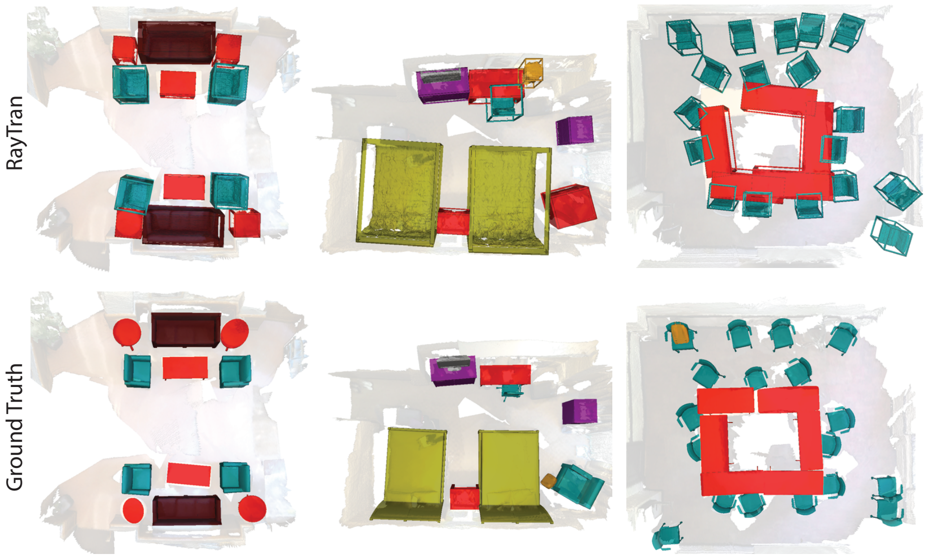

Fig. 6: **Additional qualitative Results (top-view)**: We show the 3D pose estimation and shape reconstruction outputs for 3 additional scenes. For each detected object we visualize its 3D oriented bounding box, as well as its reconstructed mesh.

frames at inference time. With 48 frames class-avg. falls by -0.4%. Worse yet, if inference were constrained to 20 frames as during training, then performance would drop by -3.3%. This highlights the value of our model's ability to input a variable number of frames.

5 Conclusions

We presented RayTran, a novel backbone architecture for 3D scene reconstruction from RGB video frames, that uses transformers for unprojecting 2D features and consolidating them into a global 3D representation. We introduced the ray-traced sparse transformer block, which enables feature sharing between the 2D and 3D network streams, in a computationally feasible way on current hardware. We use this architecture to perform 3D object reconstruction for the full scene by combining it with a DETR-style network head. Our architecture can reconstruct the whole scene in a single pass, is end-to-end trainable, and does not rely on tracking. We perform experiments on the Scan2CAD benchmark, where RayTran outperforms (1) recent state-of-the-art methods [39,34,35,15] for 3D object pose estimation from RGB videos; and (2) a strong alternative method combining Multi-view Stereo [17] with RGB-D CAD alignment [4].

References

1. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: CVPR (2008) 3
2. Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M., Schmid, C.: ViViT: A video vision transformer. In: CVPR (2021) 2, 4
3. 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) 1, 3, 4, 9, 10, 11
4. Avetisyan, A., Dai, A., Nießner, M.: End-to-end cad model retrieval and 9dof alignment in 3d scans. In: ICCV (2019) 1, 3, 4, 9, 10, 11, 14
5. Avetisyan, A., Khanova, T., Choy, C., Dash, D., Dai, A., Nießner, M.: SceneCAD: Predicting object alignments and layouts in RGB-D scans. In: ECCV (2020) 4, 10
6. Bergmann, P., Meinhardt, T., Leal-Taixe, L.: Tracking without bells and whistles. In: ICCV (2019) 3
7. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding? In: ICML (2021) 2, 4
8. Breitenstein, M.D., Reichlin, F., Leibe, B., Koller-Meier, E., Van Gool, L.: Robust tracking-by-detection using a detector confidence particle filter. In: ICCV (2009) 3
9. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020) 1, 2, 4, 5, 8, 12
10. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv:1512.03012 (2015) 9
11. Chen, Z., Tagliasacchi, A., Zhang, H.: Bsp-net: Generating compact meshes via binary space partitioning. In: CVPR (2020) 5
12. Cheng, B., Schwing, A.G., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. In: NeurIPS (2021) 4
13. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In: ECCV (2016) 5
14. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR (2017) 9
15. Danila Rukhovich, Anna Vorontsova, A.K.: ImVoxelNet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In: WACV (2022) 1, 3, 4, 10, 11, 14
16. 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. In: ICLR (2020) 4
17. Duzceker, A., Galliani, S., Vogel, C., Speciale, P., Dusmanu, M., Pollefeys, M.: Deepvideomvs: Multi-view stereo on video with recurrent spatio-temporal fusion. In: CVPR (2021) 1, 3, 9, 11, 14
18. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. TPAMI **40**(3), 611–625 (2017) 3
19. Engelmann, F., Rematas, K., Leibe, B., Ferrari, V.: From points to multi-object 3d reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4588–4597 (2021) 5
20. Fei, X., Soatto, S.: Visual-inertial object detection and mapping. In: ECCV (2018) 4

21. Frome, A., Huber, D., Kolluri, R., Bülow, T., Malik, J.: Recognizing objects in range data using regional point descriptors. In: ECCV (2004) 4
22. Girdhar, R., Fouhey, D., Rodriguez, M., Gupta, A.: Learning a predictable and generative vector representation for objects. In: ECCV (2016) 5
23. Gkioxari, G., Malik, J., Johnson, J.: Mesh R-CNN. In: ICCV (2019) 5
24. Gümeli, C., Dai, A., Nießner, M.: Roca: Robust cad model retrieval and alignment from a single image. arXiv:2112.01988 (2021) 5
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 6
26. Hu, H.N., Cai, Q.Z., Wang, D., Lin, J., Sun, M., Krahenbuhl, P., Darrell, T., Yu, F.: Joint monocular 3d vehicle detection and tracking. In: ICCV (2019) 5
27. Huang, S., Qi, S., Zhu, Y., Xiao, Y., Xu, Y., Zhu, S.C.: Holistic 3D scene parsing and reconstruction from a single RGB image. In: ECCV (2018) 5
28. Izadinia, H., Seitz, S.M.: Scene recomposition by learning-based icp. In: CVPR (2020) 4, 10
29. Izadinia, H., Shan, Q., Seitz, S.M.: Im2CAD. In: CVPR (2017) 5
30. Kundu, A., Li, Y., Rehg, J.M.: 3D-RCNN: Instance-level 3d object reconstruction via render-and-compare. In: CVPR (2018) 5
31. Kuo, W., Angelova, A., Lin, T.Y., Dai, A.: Mask2CAD: 3D shape prediction by learning to segment and retrieve. In: ECCV (2020) 3, 4, 5, 9, 10
32. Kuo, W., Angelova, A., Lin, T.Y., Dai, A.: Patch2cad: Patchwise embedding learning for in-the-wild shape retrieval from a single image. In: ICCV (2021) 5
33. Lewiner, T., Lopes, H., Vieira, A.W., Tavares, G.: Efficient implementation of marching cubes' cases with topological guarantees. J. Graphics, GPU, & Game Tools **8**(2), 1–15 (2003) 2, 8
34. 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) 1, 2, 3, 9, 10, 11, 14
35. Li, K., Rezatofighi, H., Reid, I.: MOLTR: Multiple object localization, tracking and reconstruction from monocular rgb videos. IEEE Robotics and Automation Letters **6**(2), 3341–3348 (2021) 1, 3, 10, 11, 14
36. Li, Y., Dai, A., Guibas, L., Nießner, M.: Database-assisted object retrieval for real-time 3d reconstruction. In: Computer Graphics Forum. vol. 34. Wiley Online Library (2015) 4
37. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) 10
38. Mahendran, S., Ali, H., Vidal, R.: A mixed classification-regression framework for 3d pose estimation from 2d images. arXiv:1805.03225 (2018) 5
39. Maninis, K.K., Popov, S., Niesser, M., Ferrari, V.: Vid2CAD: Cad model alignment using multi-view constraints from videos. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 1, 2, 3, 4, 9, 10, 11, 14
40. Meinhardt, T., Kirillov, A., Leal-Taixe, L., Feichtenhofer, C.: Trackformer: Multi-object tracking with transformers. arXiv (2021) 4
41. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: CVPR (2019) 5
42. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR (2017) 5
43. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular slam system. IEEE transactions on robotics (2015) 3
44. Nan, L., Xie, K., Sharf, A.: A search-classify approach for cluttered indoor scene understanding. ACM Transactions on Graphics (TOG) (2012) 4

45. Nicholson, L., Milford, M., Sünderhauf, N.: Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam. RA-L **4**(1), 1–8 (2018) 3
46. 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: CVPR (2020) 5
47. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: CVPR (2019) 5
48. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf 10
49. Pollefeys, M., Koch, R., Van Gool, L.: Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. IJCV **32**(1), 7–25 (1999) 3
50. Popov, S., Bauszat, P., Ferrari, V.: CoReNet: Coherent 3D scene reconstruction from a single RGB image. In: ECCV (2020) 5
51. Qian, S., Jin, L., Fouhey, D.F.: Associative3d: Volumetric reconstruction from sparse views. In: ECCV (2020) 3
52. Runz, M., Li, K., Tang, M., Ma, L., Kong, C., Schmidt, T., Reid, I., Agapito, L., Straub, J., Lovegrove, S., et al.: Frodo: From detections to 3d objects. In: CVPR (2020) 2, 3
53. Salas-Moreno, R.F., Newcombe, R.A., Strasdat, H., Kelly, P.H., Davison, A.J.: SLAM++: Simultaneous localisation and mapping at the level of objects. In: CVPR (2013) 4
54. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: CVPR (2016) 3, 7
55. Shan, M., Feng, Q., Jau, Y.Y., Atanasov, N.: ELLIPSDF: Joint object pose and shape optimization with a bi-level ellipsoid and signed distance function description. In: ICCV (2021) 4, 10
56. Shao, T., Xu, W., Zhou, K., Wang, J., Li, D., Guo, B.: An interactive approach to semantic modeling of indoor scenes with an RGBD camera. ACM Transactions on Graphics (TOG) (2012) 4
57. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: ICCV (2021) 4
58. Tulsiani, S., Gupta, S., Fouhey, D., Efros, A.A., Malik, J.: Factoring shape, pose, and layout from the 2d image of a 3d scene. In: CVPR (2018) 5
59. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017) 1, 4, 6, 7
60. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single RGB images. In: ECCV (2018) 5
61. Wu, C.: Towards linear-time incremental structure from motion. In: 3DV (2013) 3
62. Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B.: Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: NIPS (2016) 5

63. Xie, H., Yao, H., Zhang, S., Zhou, S., Sun, W.: Pix2vox++: multi-scale context-aware 3d object reconstruction from single and multiple images. IJCV **128**(12), 2919–2935 (2020) 5
64. Yang, S., Scherer, S.: Cubeslam: Monocular 3-d object slam. IEEE Transactions on Robotics **35**(4), 925–938 (2019) 3
65. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR (2021) 4