Sparse Multi-Object Render-and-Compare

Florian Langer
fml35@cam.ac.uk
Ignas Budvytis
ib255@cam.ac.uk
Roberto Cipolla
rc10001@cam.ac.uk

Department of Engineering
University of Cambridge
Cambridge, UK

Abstract

Reconstructing 3D shape and pose of static objects from a single image is an essential task for various industries, including robotics, augmented reality, and digital content creation. This can be done by directly predicting 3D shape in various representations [12, 28, 34] or by retrieving CAD models from a database and predicting their alignments [13, 14, 21, 22, 24]. Directly predicting 3D shapes often produces unrealistic, overly smoothed or tessellated shapes [12, 27, 33]. Retrieving CAD models ensures realistic shapes but requires robust and accurate alignment. Learning to directly predict CAD model poses from image features is challenging and inaccurate [21, 22]. Works, such as ROCA [14], compute poses from predicted normalised object coordinates which can be more accurate but are susceptible to systematic failure. SPARC [24] demonstrates that following a "render-and-compare" approach where a network iteratively improves upon its own predictions achieves accurate alignments. Nevertheless, it performs individual CAD alignment for every object detected in an image. This approach is slow when applied to many objects as the time complexity increases linearly with the number of objects and can not learn inter-object relations. Introducing a new network architecture Multi-SPARC we learn to perform CAD model alignments for multiple detected objects jointly. Compared to other single-view methods we achieve state-of-the-art performance on the challenging real-world dataset ScanNet [8]. By improving the instance alignment accuracy from 31.8% [14] to 40.3% we perform similar to state-of-the-art multi-view methods [26].

1 Introduction

Approaches to reconstructing 3D scenes from an image can be broadly split up into direct shape prediction [12, 27, 33] as well as retrieval-based methods [13, 14, 21, 22]. The issue with the former is that they struggle to reconstruct high quality shapes. Retrieval-based approaches on the other hand often have difficulty in accurately aligning CAD models to an image. Some existing works [21, 22] directly regress CAD model poses from image features. Whilst being simple such methods are often inaccurate. Other methods, such as ROCA [14], predict dense 2D to 3D correspondences and use these correspondences for computing object poses. While such approaches allow for more accurate pose estimates, the predicted correspondences are often systematically shifted, leading to a constant offset in the

Figure 1: **State-of-the-art methods for CAD model alignment from a single image.** Using normalised object coordinates ROCA's [14] alignments suffer from constant offsets. SPARC [24] produces more accurate alignments, but predicts CAD model poses individually which is slow and leads to worse predictions. In our method CAD model alignments are predicted jointly which is faster and more accurate.

alignment. Recent work [24] demonstrates that an iterative, render-and-compare approach is more accurate and robust than relying on normalised object coordinates. However, [24] perform CAD model alignment individually for every detected object which is slow at test time and can not model inter-object relations. We introduce a render-and-compare approach to deal with multiple CAD models simultaneously. For this purpose we predict bounding boxes, surface normals, depth and segmentation masks for a given input image. For every detected bounding box we initialise a CAD model in some initial pose and reproject points and surface normals sampled from the CAD model into the image plane. This information in combination with sparse information about the depth, surface normals, segmentation masks and RGB is used as the input to a Perceiver-based [19] alignment network which predicts pose updates for all CAD models jointly.

We demonstrate that learning pose alignments jointly and pre-training our network on a large number of randomly sampled synthetic scenes leads to state-of-the-art-performance on the real-world dataset ScanNet [8]. Another important observation is that our network benefits from imposing some structure on the latent space. In addition to learning pose alignments we learn classification scores indicating whether the current alignment is accurate or not. We show that we can use these classification scores to select the best alignment from different initialisations. Our system improves the instance alignment accuracy on ScanNet [8] from 31.8% [14] to 40.3%. In summary our contributions include:

- A novel render-and-compare approach which jointly predicts CAD model alignments for multiple CAD models simultaneously;
- A demonstration that synthetic pre-training on a large number of synthetic scenes achieves state-of-the-art performance on the challenging real-world dataset ScanNet [8].
- A well calibrated classification score that can be used for selecting CAD model poses from different initialisations and other tasks.

2 Related Work

Aligning CAD models to images is a form of 3D reconstruction. While there exist a large number of works that perform 3D reconstruction by directly predicting shapes in various representations [7, 9, 10, 12, 31, 33, 36], this section will focus on works that, like ours, perform 3D reconstruction by retrieving CAD models and aligning them to images. Those

works can be split along two meaningful axes: Whether they are single-shot predictions or perform iterative render-and-compare, or whether they predict object poses individually or for multiple CAD models jointly.

Single-shot alignments vs. iterative procedures. Mask2CAD [21] and Patch2CAD [22] directly predict CAD model poses by simply regressing the 6-DoF pose with a convolutional network. While this approach is very simple and fast it is not very accurate and performs poorly for unseen objects. [23] demonstrate more accurate alignments by establishing sparse 2D-3D correspondences between RGB images and rendered CAD model and use these constraints to find the pose that maximizes the silhouette overlap with an instance segmentation prediction. ROCA [14] demonstrate a more robust method by leveraging predicted depth to lift dense 2D-3D correspondences into 3D and directly optimizing for the pose that minimizes the 3D correspondence error. In contrast to these works stand approaches that iteratively update a CAD model pose. These works include [18] and [13] which learn a comparison function between the original image and the rendered CAD model. Both works maximise the learned similarity function at test time using gradient descent requiring 250 to 1000 update steps with run-times of 4 minutes and 36 seconds respectively. SPARC [24] demonstrate that render-and-compare can be harnessed more efficiently by directly learning to predict pose updates which proves to be a lot faster (2 seconds) and more robust to poor initialization. Our method works similar to SPARC [24] but we demonstrate how to apply render-and-compare to multiple objects simultaneously.

Single-object vs. multi-object. [13, 14, 21, 22, 24] all predict alignments for every CAD model individually. While [14, 21, 22] are still fast as they use the same encoder for making predictions for multiple CAD models, [13, 24] need to perform render-and-compare separately for every object which is slow at test time as the time increases linearly with the number of objects in the scene. This can be very slow for scenes with many objects. Independent of the speed all of these methods fail to model inter-object relations which are valuable when attempting to predict accurate CAD alignments.

Methods like [3, 27, 37] explicitly model inter-object relations demonstrating that these can contain valuable information for the alignment. [3, 27, 37] model object relations with a graph where nodes represent objects and edges represent their relations with each other. In comparison we allow our network to learn object relations by imposing less structure by having a dense latent space where information from different objects can attend to information regarding its own alignment and the alignment of other objects through attention.

3 Method

In this section we describe the three key steps of our method: (i) 2D object detection, instance segmentation as well as surface normal and depth estimation (Sec. 3.1), (ii) sparse input generation (Sec. 3.2) and (iii) pose update predictions (Sec. 3.3) where we iteratively repeat steps (ii) and (iii). Sec. 3.4 explains the synthetic pre-training we used.

3.1 Object Detection, Instance Segmentation, Normal and Depth Prediction

As a first step we perform 2D object detections by predicting a set of bounding boxes (BB) and object classes (see Fig. 2) using Mask-RCNN [15]. We use the same bounding boxes, object classes and CAD model retrievals as ROCA [14], although any other method could

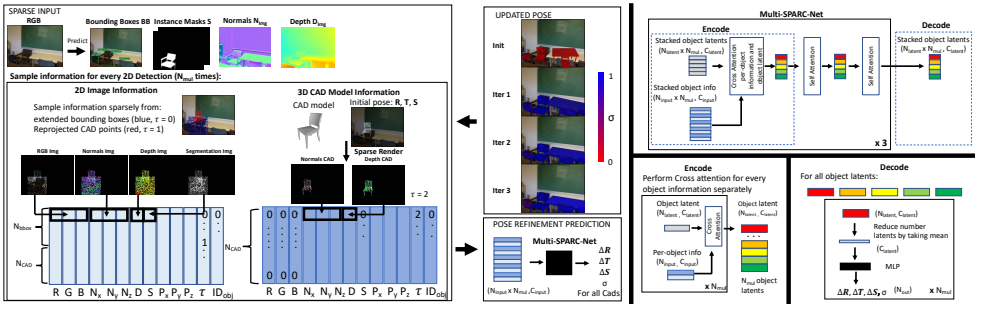

Figure 2: **Method**: Left side: For all 2D detections we sample the RGB values (RGB), surface normals (N), depth values (D) and instance segmentation mask values (S) from inside the detected bounding boxes and for pixel bearing (P_x, P_y, P_z) onto which a 3D CAD point is reprojected. CAD model information is encoded by reprojecting 3D points and surface normals into the image plane. Right side: Using Multi-SPARC-Net we encode information for each alignment separately into a latent space using cross-attention. Repeating blocks of separate cross-attention followed by self-attention layers three times we decode from each part of the latent space separately to predict pose updates $\Delta\mathbf{R}$, $\Delta\mathbf{T}$ and $\Delta\mathbf{S}$ as well as a classification score σ. Pose updates are used to iteratively refine CAD model poses and σ is used for choosing the best alignment from different rotation initialisations (see Fig. 4a).

be employed as well. Additionally, we use instance segmentation predictions (S) from [20] prompted with the detected bounding boxes. For estimating surface normals (N) and depth values (D) we follow the same training procedure as [24]. We employ a lightweight convolutional encoder-decoder architecture from [1]. The training losses are consistent with state-of-the-art works for surface normal estimation [4] and for depth estimation [5]. We use ground truth surface normals provided by [16] and ground truth depth from ScanNet [8] (for more details see the Supp. Mat.). When training the surface normal and depth estimation network, we respect the train and test split used in our evaluation.

3.2 Generating Sparse Inputs

Rather than processing full images we sample sparse image information as vectors through different image channels [24]. We sample the location of those vectors from two regions, inside the detected bounding boxes (blue points in Fig. 2) and from pixels onto which 3D CAD model points were reprojected (red points). The different input channels include their color values (RGB), surface normal (N) and depth estimates (D) as well as their instance segmentation mask value (S). We append to those vectors the corresponding pixel bearing (P_x, P_y, P_z) (to provide information on the location of the sampled values), a token τ corresponding to the type of input ($\tau = 0$ for bounding box, $\tau = 1$ for reprojected points) and the ID of the detection. For a single detection all vectors are stacked to make up the light blue block of shape $(N_{bbox} + N_{CAD}, C_{input})$ in Fig. 2. We encode the 3D CAD model information of shape (N_{CAD}, C_{input}) (dark blue block) in a similar way by sampling 3D points and corresponding surface normals from the CAD model in the current pose $\mathbf{R}, \mathbf{T}, \mathbf{S}$. When reprojecting those points into the image plane we can compute the locations of the corresponding pixel bearings and the values of their surface normal and depth. Values for the color channels (RGB) and instance segmentation (S) are filled with zeros. For the region channel we add $\tau = 2$ and also

include the detection ID. Together, both blocks of information make up all the information for a given detection which is encoded separately into the latent space. This information is sampled for all detections up to a maximum number of N_{mul} detections. If there are fewer detections than N_{mul} inputs are padded with zeros. If there are more detections, they are split up into multiple forward passes.

3.3 Pose Update Predictions

This subsection provides details on the network architecture, pose parameterisation, loss function and iterative refinement procedure.

Network Architecture. Our network architecture is built on a Perceiver network [19] with one small difference. Rather, than encoding all input information of the different detections jointly we found it beneficial to encode them separately using a shared cross-attention layer ($[N_{input}, C_{input}], [N_{latent}, C_{latent}] \rightarrow [N_{latent}, C_{latent}]$) (see Fig. 2 right side). We concatenate all encodings and apply two layers of self-attention ($[N_{mul} \cdot N_{latent}, C_{latent}] \rightarrow [N_{mul} \cdot N_{latent}, C_{latent}]$) which allows for processing information relevant to the alignment and for sharing information between the different alignments. This block of per-object cross-attention followed by two layers of self attention is repeated three times. At the decoding stage we again decode from the relevant portion of the latent space for each detection separately. For this we reduce the $[N_{\text{Latent}}, C_{\text{Latent}}]$ latent space for each object to an $[C_{\text{Latent}}]$ embedding by taking the mean over the first dimension. We map this to the desired number of output parameters $N_{\text{out}} = 11$ using an MLP. The same MLP is applied to the different portions of the latent space to produce pose updates for every detection.

Pose Parameterisation. The outputs are the updates to the current pose $(\mathbf{T}, \mathbf{R}, \mathbf{S})$. They consist of a translation update $\Delta\mathbf{T}$, a rotation update $\Delta\mathbf{R}$ and a scale update $\Delta\mathbf{S}$ as well as a classification score σ indicating whether the starting pose was already an accurate alignment or not. We parameterise $\mathbf{T}$ with polar coordinates (d, ϕ, θ) where d is the distance from the camera center and ϕ and θ parameterise a vector on the unit sphere. The updated translation $\mathbf{T}'$ is given by $\mathbf{T}' = (d \cdot \Delta d, \phi + \Delta\phi, \theta + \Delta\theta)$. Rotation is parameterised using quarternions which are transformed to a rotation matrix before making the rotation update $\mathbf{R}' = \mathbf{R} \cdot \Delta\mathbf{R}$. Finally, $\mathbf{S}$ is parameterised by three axis-aligned scaling parameters and $\mathbf{S}' = \mathbf{S} \cdot \Delta\mathbf{S}$. The updates for scale and the distance parameter d are multiplicative rather than additive. This is to ensure that the learned updates are decoupled from each other as much as possible. An additive scale update will produce different effects depending on whether the object is close and small or far away and large. In contrast, a multiplicative scale update will produce the same result. We ensure that the predicted updates are positive by applying a sigmoid function to the predicted values. Choosing polar coordinates was again motivated by the intuition that decoupled pose updates are easier to learn than coupled ones. While for euclidean coordinates a given X prediction will have a very different effect if the object is close and small or far and large, predicting updates for ϕ and θ will have the same effect regardless of the distance.

Loss function. Our loss function is comprised of two components, one for learning the CAD model alignments and one for learning the pose classifications. For learning the alignments we introduce a loss function that unifies learning translation, rotation and scale, and does not require any hyper-parameter tuning for weighing the relative strengths of different components. Our loss is simply given by the L1 distance of N_{loss} points $\mathbf{P}$ sampled from the CAD model in the ground truth pose $(\mathbf{T}_{\text{GT}}, \mathbf{R}_{\text{GT}}, \mathbf{S}_{\text{GT}})$ to the CAD model under the predicted pose $(\mathbf{T}', \mathbf{R}', \mathbf{S}')$, $L_{\text{align}} = \sum_{i=1}^{N_{\text{loss}}} |F'(\mathbf{P}_i) - F_{GT}(\mathbf{P}_i)|$, where F' and F_{GT} denote the affine transfor-

mations when applying $\mathbf{S}'$,$\mathbf{R}'$ and $\mathbf{T}'$ or $\mathbf{S}_{GT}$,$\mathbf{R}_{GT}$ and $\mathbf{T}_{GT}$ respectively. In general, poses are initialised from a large range of translations, rotations and scale to ensure that at test time the network is robust to poor detections. Consistent with previous work [21, 24], we find that it is difficult to learn rotation updates over the entire rotation space. We therefore constrain initialisations to be within an azimuthal angle of $\pm 45^\circ$ of $\mathbf{R}_{\text{GT}}$. At test time we initialise from 0°, 90°, 180° and 270° azimuthal angle and use the predicted pose classification σ to identify the correct prediction. For learning σ we use a binary cross entropy loss. A given pose is labelled correct if its translation, rotation and scale are within 20 cm, 20° and 20% respectively, $L_{\text{classifier}} = L_{\text{BCE}}(\sigma, \sigma_{\text{GT}})$. Therefore the total loss is given by $L_{\text{total}} = L_{\text{align}} + L_{\text{classifier}}$. In order to balance the training of the pose classifier we sample separate training poses which are different from the ones used for learning the pose updates (see the Supp. Mat.).

Iterative Refinements. After a given prediction at train time the next initial poses will be the updated poses based on the networks predictions. This ensures that the network learns to predict pose updates for realistic poses that it is likely to encounter at test time. After repeating this 3 times a new batch of images is initialised with objects sampled in random poses. At test time pose updates are predicted for all objects in the image which are initialised from 4 different azimuthal angles rotated 90° with respect to each other (Fig. 2 shows just one such initialisation). For each initialisation three pose updates are predicted and in a fourth iteration their classification score σ is determined. For each detection the pose with the highest classification score is returned as the final prediction (see Fig. 4a).

3.4 Synthetic Pre-training

For the synthetic pre-training we sample random objects from 3D-Future [11] in random poses and render them on-the-fly with PyTorch3D [29]. We use CAD models from 3D-Future as opposed to the CAD models from ShapeNet [6] used for our main training and evaluation as many ShapeNet models contain holes or are poorly meshed leading to artifacts when rendering surface normals. For more details see the Supp. Mat.

4 Experimental Setup

This section provides a concise overview of the dataset employed in training and testing, along with an explanation of the evaluation metrics and the selected hyperparameters.

ScanNet dataset. Following the approach of [14, 21, 22, 24, 27], we use the ScanNet25k image dataset [8] for training and testing, which includes CAD model annotations provided by [2]. This dataset comprises 20,000 training images from 1,200 training scenes and 5,000 test images from 300 distinct test scenes. Our method is trained and tested on the top 9 categories with the highest number of CAD annotations covering over 2,500 unique shapes.

Evaluation metrics. For our main evaluation we follow the original evaluation protocol established by Scan2CAD [2] which evaluates CAD model alignments on a per-scene basis. We convert predicted CAD model poses into ScanNet [8] world coordinates and, similar to [14, 24], apply 3D non-maximum suppression to remove multiple detections of identical objects from different images. For the evaluation, a CAD model prediction is deemed correct if the object class prediction is correct, the translation error is less than 20 cm, the rotation error is less than 20°, and the scale error is below 20%. We report the percentage of correct alignments for each class individually as well as the overall instance alignment accuracy for all predictions.

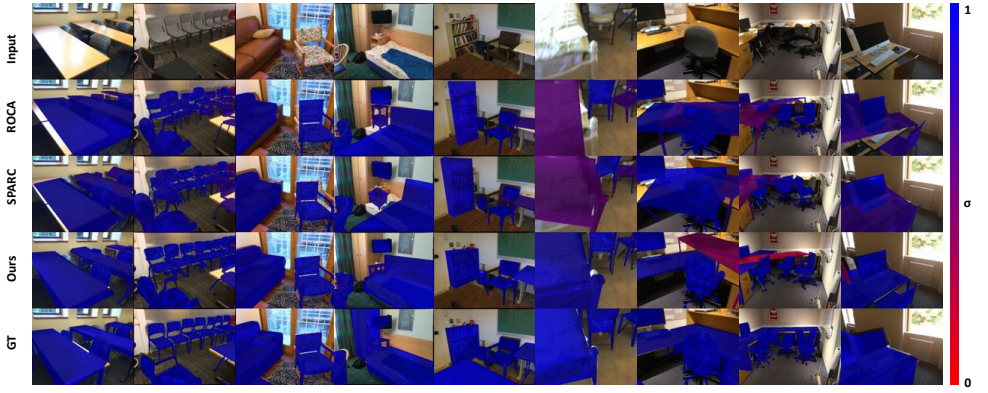

Figure 3: **Qualitative comparison.** Particularly for multiple objects close to each other our alignments are more accurate than existing methods (column 1 - 5). Due to the synthetic pre-training, our network can even work from very challenging viewpoints (column 6). Furthermore, our learned 3D classification score allows the network to identify potentially bad alignments (column 7 - 8). Our network struggles to correctly classify display orientations leading to poor performance on that class (column 9).

In addition to the per-scene alignments we evaluate per-image alignments. For this purpose we reproject CAD models in GT poses into the individual camera frames. Note that for each camera frame only GT CAD models whose center is reprojected into the camera view are considered. For every predicted CAD alignment we find the associated GT CAD model by computing the IoU of the 2D bounding boxes and finding that GT CAD model of the same category with maximum IoU. In order to avoid penalising for objects that are not visible due to occlusion we only consider GT objects for which at least 50% of pixel have the rendered depth value within 30 cm of the GT sensor depth value. Similar to the per-scene metric we evaluate the alignment accuracy by computing the percentage of predictions whose errors for rotation, translation and scale are within the same thresholds as above. Additionally we compute AP^{mesh} introduced by [12]. It is defined as the mean area under the per-category precision-recall curves for F^{ρ} at different thresholds. The F^{ρ} score is the harmonic mean of the fraction of points sampled from the predicted aligned CAD model that are within ρ of a point sampled from the GT aligned CAD model and the fraction of points sampled from the GT CAD model within ρ of a point sampled from the predicted CAD model. We evaluate AP50, which considers a prediction to be correct if $F^{\rho} > 0.5$, as well as AP mean which takes the average across the ten AP scores AP50, AP55,...,AP95 sampled in regular intervals.

Hyperparameters. For our inputs we sample $N_{bbox} = 2000$ pixels inside the predicted bounding box which is uniformly extended by 10% and use $N_{CAD} = 500$ points from the CAD model. $N_{input} = (N_{bbox} + 2N_{CAD})$ and $C_{input} = 13$. We set the number of latents $N_{latent} = 80$ where each latent has $C_{latent} = 256$ channels. We choose $N_{mul} = 5$ which means that a maximum of 5 CAD models are processed jointly. If an image contains more than 5 detections the detections are split into multiple blocks. We show in the Supp. Mat. that we achieve similar results with larger numbers of N_{mul}. We use batches of 20 images and use the Lamb optimisier [35] with learning rate set to 0.001. We sample $N_{loss} = 1000$ points for computing the loss. Our model is pretrained on 10 M rendered images containing between 1 and 4 CAD models in random poses.

	Method	bathtub	bed	bin	bkshlf	cabinet	chair	display	sofa	table	**class**	**instance**	time [ms]
	Number of Instances #	120	70	232	212	260	1093	191	113	553	9	2844	-
Ablation experiments													
Joint Encoding and Decoding	separate encoding - joint decoding	17.5	21.4	26.7	9.9	15.0	45.7	3.1	29.2	17.5	20.7	27.9	864
	joint encoding - separate decoding	18.3	34.3	36.6	12.7	14.2	52.8	4.7	25.7	17.5	24.1	31.9	656
Single vs Multi Pre-training vs. No Pre-trainig	single-object - no pre-training	20.8	24.3	39.2	12.7	22.7	57.5	2.1	24.8	19.2	24.8	34.6	2320
	multi-object - no pre-training	20.0	28.6	40.1	13.7	20.4	59.9	0.5	36.3	23.0	26.9	36.7	864
	single-object - pre-training	28.3	35.7	36.6	20.3	23.1	61.4	4.8	37.2	23.5	30.2	38.7	2320
	multi-object - pre-training	25.8	34.3	44.8	17.0	19.2	64.8	5.8	35.4	25.5	30.3	40.3	864
Sparser and Faster	$N_{bbox} = 200$, $N_{CAD} = 200$ - joint encoding	25.0	34.3	33.6	14.2	17.7	56.6	2.6	35.4	21.2	26.7	34.8	480
	$N_{bbox} = 50$, $N_{CAD} = 50$ - joint encoding	14.2	25.7	31.0	9.9	18.1	55.0	7.3	29.2	22.8	23.7	33.4	**448**
Learned Classification Score	2D confidence	27.5	31.4	45.3	16.0	20.4	60.6	5.8	38.9	25.1	30.1	38.8	816
	3D classification	25.8	34.3	44.8	17.0	19.2	64.8	5.8	35.4	25.5	30.3	40.3	864
Comparison to other methods - per-scene evaluation													
Single-view	Total3D-ODN [27]	10.0	2.9	16.8	2.8	4.2	14.4	13.1	5.3	6.7	8.5	10.4	-
	Mask2CAD-b5 [21]	7.5	2.9	24.6	1.4	5.0	29.9	13.1	5.3	5.6	10.6	16.7	60
	ROCA [14]	20.8	8.6	26.3	9.0	13.1	39.9	**24.6**	10.6	12.7	18.4	25.0	53
	SPARC [24]	**25.8**	25.7	24.6	14.2	**20.8**	51.5	17.8	28.3	15.4	24.9	31.8	1925
	Ours	**25.8**	**34.3**	**44.8**	**17.0**	19.2	**64.8**	5.8	**35.4**	**25.5**	**30.3**	**40.3**	864
Multi-view	Vid2CAD [26]	27.5	35.7	45.7	9.9	21.5	63.4	33.0	24.8	25.8	31.9	41.0	2500

Table 1: **Alignment Accuracy on ScanNet** [2, 8] in % for the per-scene evaluation in comparison to the state-of-the-art. Bolded numbers denote the highest accuracy for the single-view methods. Times are for reconstructing a scene containing 5 objects. The yellow row highlights the reference for comparing ablations for "joint encoding and decoding" as well as the "sparser and faster" experiments for which no pre-training was performed. The orange row are our main results.

Implementation Details. All code is implemented in PyTorch. Pre-training our main model takes 6 days on a single TitanXp. Finetuning on ScanNet25k for 500 epochs takes 2 days.

5 Results

This section explains our qualitative and quantitative results. We first ablate major design choices in the network architecture and training procedure and subsequently compare our method to the state-of-the-art. If not stated otherwise numbers in the following refer to the overall instance alignment accuracy of all objects on ScanNet [8].

Separate Encoding and Decoding. When performing multi-CAD model alignment with a transformer-based [32] architecture, naively one would simply concatenate all inputs, marking information for different alignments with different tokens, and hoping that the network will learn to regress all pose updates jointly. The first two rows in Tab. 1 show results for the experiments where we perform joint decoding or joint encoding. For the former we reduce all latents $[N_{mul} \cdot N_{latent}, C_{latent}] \rightarrow [C_{latent}]$ by taking the mean over the first dimension and then learning an MLP to map to $N_{mul} \cdot N_{out}$ directly. For the latter we have one large cross attention that maps from all the concatenated inputs to all latents $([N_{mul} \cdot N_{input}, C_{input}], [N_{mul} \cdot N_{latent}, C_{latent}] \rightarrow [N_{mul} \cdot N_{latent}, C_{latent}])$. Comparing the instance alignment accuracy 27.9% and 31.9% to the alignment accuracy for the multi-object results without pre-training 36.7% we find that both separate encoding and separate decoding are crucial for good alignments, with separate decodings being even more important. The intuition behind this is that it is not easy for the network to learn to associate input information from different CAD models to the correct output values and encoding and decoding separately helps with this.

Single vs. Multi-object and Pre-training vs. No Pre-training. Our experiments show that performing CAD model alignments jointly leads to slightly more accurate alignments (36.7% vs. 34.6% without pre-training, 40.3% vs. 38.7% with pre-training). Reasons why learning joint-alignments does not help even more may include noise in the annotation data, making if difficult to learn exact relations, as well as a higher chance of overfitting to entire

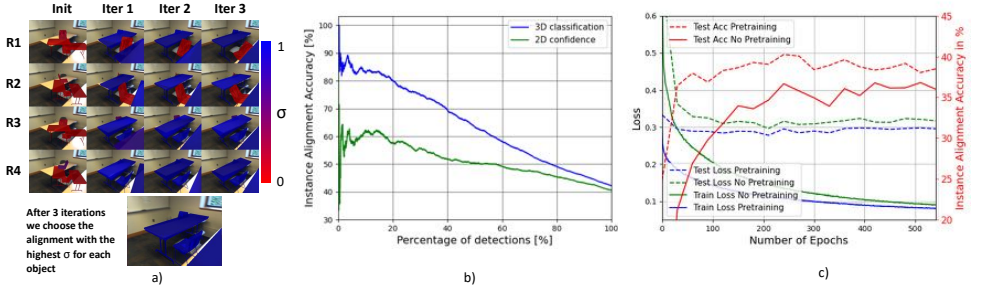

Figure 4: **Pose selection, calibration and loss functions.** a) We use the predicted classification score to select the final object predictions from 4 different rotation initialisations. b) The classification score is also used in the ScanNet evaluation to filter out duplicate predictions. Compared to the 2D confidence scores (green) from [14] our 3D classification score (blue) is significantly better calibrated. c) Synthetic pre-training leads to lower losses during training and testing as well as a higher instance alignment accuracy on the test set.

scenes as opposed to single alignments. When comparing results with and without synthetic pre-training we find significant improvement of 4%. This indicates that even training on a different set of CAD models synthetically rendered in random poses provides useful training signals that transfer to real images. Inspecting Fig. 4c we find that the pre-trained model achieves both a lower train and test loss leading to a higher instance alignment accuracy on the test set.

Sparser and Faster. Another advantage of performing alignments for multiple CAD models jointly as opposed to in sequence is that it is a lot faster. The times in Tab. 1 include the time for processing the input data (23 ms, for the main network architecture and inputs in row 4)) as well as a forward pass through the network (31 ms). These steps have to be repeated four times for the refinement procedure (3 refinement + 1 final classification score) from four different initialisations (see Fig. 4a) leading to a total time of $4 \times 4 \times (23 + 31) = 864$ ms. By processing very sparse inputs i.e. $N_{bbox} = 200$ and $N_{CAD} = 200$, reducing the number of latents $N_{latent} = 40$ and encoding input information jointly, we can reduce both the time for processing the inputs (16 ms) as well as the forward pass (14 ms) and almost halve the total run-time to 480 ms. If not initialised from four different rotations (as would be realistic for example in a video setting where the rough object rotation is known from previous frames) this approaches the speed of single-shot methods while being considerably more accurate. Interestingly, this network variant is more accurate than the one encoding the full inputs jointly in the second row. This may indicate that it is easier for the network to learn to separate information for multiple alignments when presented with fewer inputs. Row 8 shows results for even sparser inputs, resulting in further small gains in speed.

Learned classification score. Rather than just predicting pose updates we also learn classification scores indicating whether a given alignment is accurate or not. We use these learned classification scores to select the best alignment from multiple rotation initialisations (see Fig. 4a) as well as to select from multiple predictions of the same object from different images in the Scan2CAD [2] evaluation. We compare to the 2D detection confidence from ROCA [14] and note a small improvement (40.3% compared to 38.8%). More importantly, plotting the mean accuracy of the predictions sorted by the confidence we find that our 3D classification score is significantly better calibrated (see Fig. 4b).

Comparison to other methods - per-scene evaluation. We compare our method to other

		AP^{mesh}						Alignment Accuracy	
		AP50	APmean	AP50	APmean	AP50	APmean	class	instance
	ρ	0.3	0.3	0.5	0.5	0.7	0.7	-	-
ROCA [14]		1.8	0.4	10.8	3.0	20.3	7.1	16.1	18.4
SPARC [24]		2.4	0.5	9.8	3.0	19.1	7.0	15.9	17.4
Ours		**11.6**	**3.4**	**27.0**	**11.5**	**36.4**	**18.7**	**28.1**	**31.3**

Table 2: **Per-image alignment accuracy and AP^{mesh} score on ScanNet [8].** Both AP scores and alignment accuracies are reported in %. The ρ value controls the threshold for computing the F1 score in the AP calculation. Smaller ρ values require points sampled from the predicted aligned CAD model and the GT aligned CAD model to be closer together and therefore more accurate poses. Before computing the F1 score both CAD models are re-scaled isotropically such that the longest side of the 3D bounding box of the GT CAD model is equal to 10. Therefore for a typical object of maximum width and height equal to 1 m $\rho = 0.5$ requires points sampled from the predicted CAD model to be within 5 cm of the GT CAD model and vice versa.

state-of-the-art CAD model alignment procedures [14, 21, 24, 27]. Quantitatively comparing against those methods we find that we improve significantly upon the instance alignment accuracy from 31.8% to 40.3% and the class mean accuracy from 24.9% to 30.3%. We also improve in most categories with the notable exception of displays. Here our learned classification score struggles to distinguish between front and back-facing displays which look very similar when only sparse pixels are provided (see Fig. 3 last column).

Comparison to other methods - per-image evaluation. The advantages of our method compared to previous methods are even more pronounced on the per-image evaluation then they were on the per-scene evaluation (see Tab. 2). The class and instance alignment accuracy almost double compared to previous methods (28.1% vs. 16.1% and 31.3% vs. 18.4%). AP50 and APmean show even greater relative improvements, e.g. at $\rho = 0.5$ AP50 improves from 10.8% to 27.0% and APmean improves from 3.0% to 11.5%. The reason why the improvements of our method compared to the previous ones are even more pronounced on the per-image compared to the per-scene evaluation is that the per-scene evaluation requires only one very accurate prediction for each object from any frame whereas the per-image evaluation has a high number of challenging viewpoints. Here both the multi-object predictions as well as the synthetic pre-training significantly increase the accuracy of the predictions.

6 Conclusion

We introduced a novel render-and-compare approach that jointly aligns multiple CAD models to objects in an image. This provides advantages for both speed and accuracy at test time, improving the run-time by a factor of up to 5 and improving the instance alignment accuracy on ScanNet [8] from 31.8% to 40.3%. We demonstrate that some of this improvement stems from pre-training our network on a large number of random synthetic scenes. The fact that those scenes contain objects different to the ones the network is tested on highlights the ability of our render-and-compare approach to generalise. Furthermore, we learn to predict not just pose updates but also classification scores that can be used for selecting a final pose from different candidates. In the future we would like to extend render-and-compare to multi-view scenarios as well as using larger foundational models in a render-and-compare setting to reconstruct 3D scenes.

References

[1] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. *arXiv preprint arXiv:1812.11941*, 2018.

[2] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[3] Armen Avetisyan, Tatiana Khanova, Christopher Choy, Denver Dash, Angela Dai, and Matthias Nießner. Scenecad: Predicting object alignments and layouts in rgb-d scans. In *The European Conference on Computer Vision (ECCV)*, August 2020.

[4] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Estimating and exploiting the aleatoric uncertainty in surface normal estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13137–13146, 2021.

[5] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Multi-view depth estimation by fusing single-view depth probability with multi-view geometry. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2842–2851, June 2022.

[6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Virtual, June 2020.

[8] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[9] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. CvxNet: Learnable Convex Decomposition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Virtual, June 2020.

[10] Haoqiang Fan, Hao Su, and Leonidas Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, USA, July 2017.

[11] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, pages 1–25, 2021.

[12] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *Proc. IEEE Int. Conf. on Computer Vision*, Seoul, Korea, October 2019.

[13] Alexander Grabner, Yaming Wang, Peizhao Zhang, Peihong Guo, Tong Xiao, Peter Vajda, Peter M. Roth, and Vincent Lepetit. Geometric Correspondence Fields: Learned Differentiable Rendering for 3D Pose Refinement in the Wild. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[14] Can Gümeli, Angela Dai, and Matthias Nießner. Roca: Robust cad model retrieval and alignment from a single image. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2022.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[16] Jingwei Huang, Yichao Zhou, Thomas Funkhouser, and Leonidas J Guibas. Framenet: Learning local canonical frames of 3d surfaces from a single rgb image. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[17] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. Manifoldplus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*, 2020.

[18] Hamid Izadinia, Qi Shan, and Steven M. Seitz. IM2CAD. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, USA, July 2017.

[19] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4651–4664. PMLR, 18–24 Jul 2021.

[20] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023.

[21] Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. Mask2CAD: 3D Shape Prediction by Learning to Segment and Retrieve. In *Proc. 16th European Conference on Computer Vision*, Glasgow, UK, August 2020.

[22] Weicheng Kuo, Anelia Angelova, Tsung-yi Lin, and Angela Dai. Patch2CAD: Patchwise Embedding Learning for In-the-Wild Shape Retrieval from a Single Image. In *Proc. IEEE Int. Conf. on Computer Vision*, Montreal (Virtual), October 2021.

[23] F. Langer, I. Budvytis, and R. Cipolla. Leveraging geometry for shape estimation from a single rgb image. In *Proc. British Machine Vision Conference*, (Virtual), November 2021.

[24] F. Langer, G. Bae, I. Budvytis, and R. Cipolla. Sparc: Sparse render-and-compare for cad model alignment in a single rgb image. In *Proc. British Machine Vision Conference*, London, November 2022.

[25] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019.

[26] Kevis-Kokitsi Maninis, Stefan Popov, Matthias Nießner, and Vittorio Ferrari. Vid2cad: Cad model alignment using multi-view constraints from videos. *arXiv*, 2020.

[27] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes from a Single Image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Virtual, June 2020.

[28] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, June 2019.

[29] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[30] Leslie Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *ICLR*, 2018.

[31] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives, 2018.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[33] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *Proc. 15th European Conference on Computer Vision*, Munich, Germany, September 2018.

[34] Chao-Yuan Wu, Justin Johnson, Jitendra Malik, Christoph Feichtenhofer, and Georgia Gkioxari. Multiview compressive coding for 3D reconstruction. *arXiv:2301.08247*, 2023.

[35] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020.

[36] Wei Zeng, Sezer Karaoglu, and Theo Gevers. Inferring Point Clouds from Single Monocular Images by Depth Intermediation. *arXiv*, 2020.

[37] Cheng Zhang, Zhaopeng Cui, Yinda Zhang, Bing Zeng, Marc Pollefeys, and Shuaicheng Liu. Holistic 3d scene understanding from a single image with implicit representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8833–8842, June 2021.

Supplementary Material

Here we provide additional information to our main work. Sec. A provides information for the training of the surface normal and depth estimation networks. In Sec. B we explain in detail what training examples are used for training Multi-SPARC-Net to predict pose updates and classification scores. Sec. C provides ablation experiments for the maximum number of joint predictions and the size of the latent space. Sec. D explains what CAD models were used for the synthetic pre-training. Finally, we highlight the major points of the provided video showing qualitative results on ScanNet in Sec. E.

A Details for Training Surface Normal and Depth Networks

We utilize a lightweight convolutional encoder-decoder architecture [1] to estimate both surface normals ($\mathbf{N}_{\text{Img}}$) and depth ($\mathbf{D}_{\text{Img}}$). The per-pixel probability distribution for each task is predicted, and the network is trained by minimizing the negative log-likelihood (NLL) of the ground truth. Learning parameters of probability distributions allows the networks to predict high uncertainty around object edges where GT annotations can often be wrong. This improves the quality of training [4]. The distribution for surface normals is parameterized using the Angular vonMF distribution proposed in [4], while the depth distribution is parameterized with a Gaussian distribution. After training, we only consider the predicted mean values, discarding the uncertainty. The ground truth surface normals are provided by [16], and the ground truth depth is obtained from ScanNet [8], following the train/test split. For depth estimation, we train the network on the available two million train images, while for surface normals, we train on the annotated images provided by [16] within the train image set, resulting in approximately 200K train images. Both networks are trained for ten epochs using the AdamW optimizer [25], and the learning rate is scheduled using the 1cycle policy [30] with $lr_{\max} = 3.5 \times 10^{-4}$ (same as [5]). A batch size of four is used for training both surface normals and depth networks. The steps described are consistent with the training protocol of [24].

B Joint Training Pose Updates and Pose Classifier

We train Multi-SPARC-Net to predict both pose updates as well as a classification score σ indicating whether the initial pose is correct. Similar to our evaluation a pose is classified to be correct if its translation $\Delta\mathbf{T}$ is within 20 cm, its rotation $\Delta\mathbf{R}$ within 20° and its scale $\Delta\mathbf{S}$ within 20% of the ground truth values. If we simply trained the network to predict classification scores from the initialisation used for learning pose updates, the classification scores would be heavily biased towards correct alignments as even after just a single CAD pose update the vast majority of poses would classify as correct. We therefore sample separate examples for training the pose classifier. For these separate examples no loss is backpropagated for the predicted pose updates (similar no loss is backpropagated from the classification score for the examples used to train the pose updates). Examples for training the pose classifier versus the pose updates are sampled with a ratio of 1:3.

Sampling poses for training the pose classifier. Those examples are sampled from the five different regions stated in Tab. 3. Here $\Delta\mathbf{R}$, $\Delta\mathbf{T}$ and $\Delta\mathbf{S}$ denote the maximum bounds

Region	$\Delta\mathbf{R}$ [tilt, azim, elev] in [°]	$\Delta\mathbf{T}$ [cm]	$\Delta\mathbf{S}$[%]	Discrete Rotation	Sampling Frequency
Region 1	[7,10,10]	13	13	False	0.4
Region 2	[7,10,10]	13	13	True	0.2
Region 3	[7,45,20]	30	30	False	0.2
Region 4	[7,45,20]	30	30	True	0.1
Region 5	[20,45,20]	60	60	True	0.1

Table 3: **Different pose regions for drawing samples when learning to classify poses.** $\Delta\mathbf{R}$, $\Delta\mathbf{T}$ and $\Delta\mathbf{S}$ denote the maximum bounds from the ground truth values in which range rotation $\mathbf{R}$, translation $\mathbf{T}$ and scale $\mathbf{S}$ are sampled uniformly. When discrete rotation is set to True the rotation is randomly rotated (with equal probability) by 0°, 90°, 180° and 270° around the vertical. The sampling frequencies denote the relative frequency for sampling from the different regions.

from the ground truth values in which range rotation $\mathbf{R}$, translation $\mathbf{T}$ and scale $\mathbf{S}$ are sampled uniformly. Note that $\Delta\mathbf{R}$ denotes the difference to the ground truth rotation in terms of Euler angles for tilt, azimuthal and elevation angle (in order). $\Delta\mathbf{T}$ denotes the maximum difference to the ground truth translation in cm for x, y and z component. $\Delta\mathbf{S}$ indicates the maximum difference in % to the ground truth scale values in the three axes. Discrete rotation implies that the rotation is randomly rotated (with equal probability) by 0°, 90°, 180° and 270° around the vertical. This is to ensure that the network is exposed to examples with the correct translation and scale but with a wrong rotation which it will encounter at test time as we use the classification score to select the final alignment from different rotation initialisation. Note that the symmetry of objects is taken into account when determining if a pose should be classified as correct or not. Poses from these regions are sampled with frequency as indicated in the last column in Tab. 3. While the exact numbers in the sampling regions above do not matter, it is important to roughly balance the number of correct and incorrect poses and to ensure that the poses the network is likely to encounter at test time are covered in the training examples.

Sampling poses for training the pose updates. For learning the pose updates we sample the initial pose as follows. $\mathbf{T}$ is sampled by uniformly sampling a point within the predicted bounding box and then lifting that point into 3D by providing a z value sampled $z \sim \text{Uniform}(1,5)$ in metres. The scale $\mathbf{S}$ is sampled uniformly within the range of the minimum object scale and maximum object scale for all CAD model alignments of this category on ScanNet [8] by [2]. Finally, $\mathbf{R}$ is sampled uniformly within 10° tilt, 45° azimuthal and 20° elevation angle of the ground truth rotation.

C Simultaneous Prediction for Larger Number of CAD models

In Tab. 4 we show results when varying the maximum number of joint predictions N_{mul} and the number of latents N_{latent} dedicated to each alignment. Here we observe that compared to our main setup (highlighted in yellow) we can reduce the number of latents by a factor of two while still achieving very similar accuracies. Further, we note that increasing the number of joint alignments also achieves similar results. This brings extra advantages for speed when reconstructing scenes containing many objects.

Method	bathtub	bed	bin	bkshlf	cabinet	chair	display	sofa	table	**class**	**instance**	time [ms]
Number of Instances #	120	70	232	212	260	1093	191	113	553	9	2844	-
$N_{\text{mul}} = 5$, $N_{\text{latent}} = 40$	20.0	24.3	41.4	12.7	16.2	60.2	3.7	28.3	23.7	25.6	36.4	848
$N_{\text{mul}} = 5$, $N_{\text{latent}} = 80$	20.0	28.6	40.1	13.7	20.4	59.9	0.5	36.3	23.0	26.9	36.7	864
$N_{\text{mul}} = 10$, $N_{\text{latent}} = 20$	20.0	25.7	40.1	11.8	20.4	60.7	4.7	29.2	19.5	25.8	36.1	944
$N_{\text{mul}} = 10$, $N_{\text{latent}} = 40$	16.7	27.1	41.8	13.7	18.8	57.8	2.6	31.9	21.5	25.8	35.4	960

Table 4: **Ablation.** Alignment Accuracy on ScanNet [2, 8] when varying the number of CAD models N_{mul} for which predictions are made jointly and the number of latents N_{latent} dedicated to the processing of the information for each alignment. Times are for reconstructing a scene containing containing 5 or 10 objects respectively. Note that doubling the number of objects only leads to a marginal increase in time. The yellow row highlights the main setup for which no pre-training was performed from the results table in the paper.

D Artifacts Rendering ShapeNet Normals

For the synthetic pre-training we render CAD models in a large number of random poses and train Multi-SPARC-Net on those poses. Objects in the scenes from ScanNet [8] were annotated by [2] with CAD models from ShapeNet [6]. Those CAD models are used for the main training and evaluation. However, when rendering ShapeNet [6] CAD models with PyTorch3D [29] we get the semi-random patterns for surface normals as seen in Fig. 5a. This is because the simple Normal Shader we implemented in PyTorch3D [29] identifies which CAD model face is rendered into a given pixel and then interpolates the per-vertex surface normal value from the three vertices of the given face. The issue with rendering the original ShapeNet [6] models is that many of them are not closed or contain doubly-meshed faces, meaning that the mesh contains two identical faces with the order of two vertices swapped such that their surface normals will point in opposite directions. Rendering these means that front or back-facing surface normals will be selected at random for the interpolation, leading to the wrong surface normal renderings in Fig. 5a. We try to make ShapeNet [6] CAD models watertight with consistently oriented faces using [17]. While this allows us now to render CAD normals (Fig 5b) the generated surface normal renders sometimes still suffer from noise when the procedure for making the CAD models watertight did not succeed. Further, making the models watertight using [17] increases the median number of vertices from just 600 to 130 K which makes the CAD models impractical for us to use as it massively increases the rendering time. Instead we choose to perform our synthetic training on CAD models from 3D Future [11] which are already watertight and yield correct surface normals when rendering in PyTorch3D [29] (see Fig. 5c).

E Video with Qualitative Visualisation

We provide a video comparing our predictions qualitatively to ROCA [14] and SPARC [24] on ScanNet [8] (`https://www.youtube.com/watch?v=NtOU5BOmagw`). Note that for our predictions the color indicates the value of the learned classification score. For ROCA [14] and SPARC [24] the color indicates the value of the 2D detection score which is not updated based on the refinement for SPARC [24]. As explained in Sec. 3.3 of the main paper we initialise CAD models from four different rotations. In the video we only show those alignments corresponding to that rotation initialisation which led to the highest classification score after three refinements. This is why some of the initialisation between SPARC [24] and

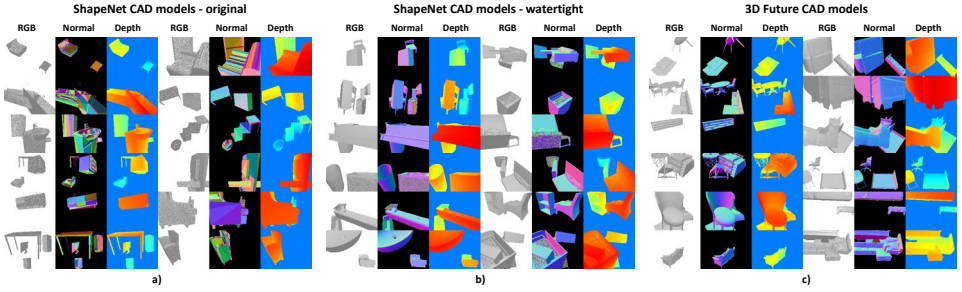

Figure 5: **Comparison when rendering ShapeNet [6] CAD models and 3D Future [11] CAD models using PyTorch3d [29].** We find that rendering normals for the original (a) as well as the processed watertight ShapeNet [6] CAD models (b) results in artifacts and therefore use CAD models from 3D Future [11] instead which result in correct renders (c).

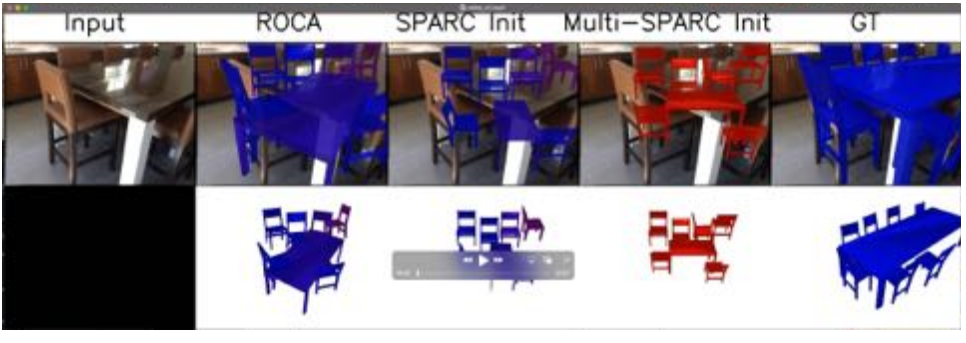

Figure 6: **Video with Qualitative Visualisations.**

ours are different. However, apart from selecting the rotation initialisation, the initialisations between SPARC [24] and ours are the same.

While ROCA's [14] predictions are often inaccurate, suffering from displacements and wrong scale predictions, we find that our predictions are accurate. Particularly, they are are also more accurate than SPARC's [24] predictions, matching object outlines more closely due to the synthetic pretraining and yielding more accurate alignments for objects in close proximity to each other due to the mulit-object training and predictions. Interestingly, we find that Multi-SPARC-Net learns to rotate objects by more than 90° around the vertical over multiple refinements even though at train time it only ever learned to predict updates that were less than 45° away from the correct rotation.