SPARC: Sparse Render-and-Compare for CAD model alignment in a single RGB Image

Florian Langer
fml35@cam.ac.uk
Gwangbin Bae
gb585@cam.ac.uk
Ignas Budvytis
ib255@cam.ac.uk
Roberto Cipolla
rc10001@cam.ac.uk

Department of Engineering
University of Cambridge
Cambridge, UK

Abstract

Estimating 3D shapes and poses of static objects from a single image has important applications for robotics, augmented reality and digital content creation. Often this is done through direct mesh predictions [14, 27, 35] which produces unrealistic, overly tessellated shapes or by formulating shape prediction as a retrieval task followed by CAD model alignment [15, 16, 22, 23]. Directly predicting CAD model poses from 2D image features is difficult and inaccurate [22, 23]. Some works, such as ROCA [16], regress normalised object coordinates and use those for computing poses. While this can produce more accurate pose estimates, predicting normalised object coordinates is susceptible to systematic failure. Leveraging efficient transformer architectures [19] we demonstrate that a sparse, iterative, render-and-compare approach is more accurate and robust than relying on normalised object coordinates. For this we combine 2D image information including sparse depth and surface normal values which we estimate directly from the image with 3D CAD model information in early fusion. In particular, we reproject points sampled from the CAD model in an initial, random pose and compute their depth and surface normal values. This combined information is the input to a pose prediction network, SPARC-Net, which we train to predict a 9 DoF CAD model pose update. The CAD model is reprojected again and the next pose update is predicted. Our alignment procedure converges after just 3 iterations, improving the state-of-the-art performance on the challenging real-world dataset ScanNet [9] from 25.0% [16] to 31.8% instance alignment accuracy. Code will be released under https://github.com/florianlanger/SPARC.

1 Introduction

Previous work on shape and pose prediction can be classified into two different types of methods relying either on shape generation [14, 27, 35] or shape retrieval [15, 16, 22, 23]. Generative approaches usually struggle to produce realistic object shapes. For methods relying on shape retrieval one of the key challenges is to align the retrieved CAD model to the object detected in the image [15, 16, 22, 23]. Many existing approaches directly regress object poses from the 2D features of the image [12, 22, 23, 25]. However, this produces

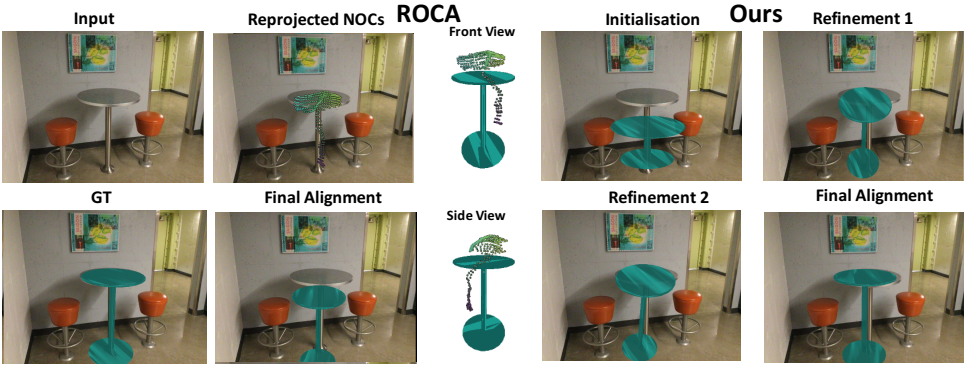

Figure 1: **Comparison of our CAD model alignment approach to ROCA [16].** For each pixel of the detected object ROCA predicts the 3D normalised object coordinates (NOCs) in a canonical, normalised frame. However, those predictions are susceptible to systematic offsets (see Front View and Side View). While the reprojected NOCs match the image, the corresponding CAD model alignment does not. Our approach in contrast reprojects points and surface normals sampled from the CAD model in an initial pose into the image and uses those to predict pose updates. By iteratively updating the pose and reprojecting our system achieves precise CAD model alignments.

approximate poses rather than accurate alignments. ROCA [16] follows a more geometric approach in which they predict normalised object coordinates (NOCs) [33], dense correspondences from 2D pixel to 3D points in a canonical object space, that are used to compute the object pose. The fundamental issue with learning NOCs is that it is unclear how different shapes should be registered with each other. This means that the NOCs do not generalise well between CAD models and predicting them often fails with a systematic offset leading to a displacement in the final alignment (see Figure 1). Rather than using NOCs we propose a sparse, render-and-compare approach. We combine 2D image information including sparse depth and surface normals along with RGB colors with 3D CAD model information. Specifically, we initialise a CAD model in a generic pose and reproject points and surface normals sampled uniformly from its surface onto the image plane. This combined information is used by our pose update prediction network, SPARC-Net, to estimate a 9-DoF pose update. After adding the predicted pose update to the initial pose we reproject the CAD model again and estimate the next pose update step. Repeating this procedure we obtain the final pose in just three iterations.

Having access to estimated normal and depth values from the image and reprojected normals and depth from the CAD model allows the network to evaluate the current pose and predict an accurate pose update by easily comparing observed (image) and projected (CAD model) information. This is in contrast to other approaches that do not make use of any shape information when predicting object poses [22, 23] or that rely on shape encodings [16] which seems to be a more difficult learning task than using render-and-compare. Note also that because of the render-and-compare our approach does not require the ability to register different CAD models with each other nor for the network to memorise 3D shapes.

We choose the Perceiver [19] architecture over traditional CNNs for the pose prediction network as Perceivers [19] allow for an efficient processing of the sparse object reprojection as they do not require a full, image-sized input. Also Perceivers [19] have linear time

and memory complexity in terms of input as opposed to traditional transformers [34]. Not only do sparse inputs reduce the memory and network complexity, but we also show that they improve the alignment accuracy by avoiding overfitting. Using our approach we improve upon the state-of-the art performance on the challenging real-world dataset ScanNet [9] from 25.0% instance alignment accuracy to 31.8% instance alignment accuracy.

Our contributions include:

- a **novel, sparse, render-and-compare method** that achieves state-of-the-art results in CAD model alignment from a single RGB image
- a **demonstration of** how 3D CAD model information and 2D image information can be effectively combined with **early-fusion**
- **highlighting that sparse inputs improve alignment accuracy** while at the same time **significantly decreasing memory and compute complexity**.

2 Related Work

In this section we discuss relevant works including shape estimation from a single image, CAD model retrieval and alignment and efficient transformer architectures.

Shape estimation from a single RGB image. Most works on shape estimation from a single image are structured such as to use an RGB image as input to a neural network and directly predict a 3D shape using some particular representation. Various representations have been explored ranging from voxels [8] to point clouds [13, 38], meshes [14, 27, 28, 35], packed spheres [13], binary space partitioning [7], convex polytopes [10], signed distance fields [29] to other implicit representations [26]. However, this is a difficult learning task and regardless of the representation chosen most approaches struggle to predict realistic shapes. Because of these issues we use a retrieval-based method.

CAD model retrieval and alignment. Rather than generating shapes a second class of approaches estimates 3D shapes by retrieving from large-scale CAD model databases [6]. The large number of available CAD models ensures that appropriate CAD models can be retrieved which can reconstruct the scene in a clean and compact representation that can be easily consumed for downstream applications. Existing approaches differ significantly in their procedures for aligning the retrieved CAD models to the image. Mask2CAD [22] as well as [23, 25] simply regress the object pose from image features. However, this produces inaccurate pose estimates and requires the network to memorize every CAD model shape, consequently performing poorly for unseen CAD models. [24] explicitly leverages the geometry of the retrieved CAD model by estimating 2D-3D keypoint matches but depend on accurate segmentation masks and require exact correspondence between CAD model and the object observed in the image. ROCA [16] learns dense correspondences between 2D pixels and 3D object coordinates in a normalised, canonical frame and use those for computing the object pose. While these dense correspondences are more robust than sparse ones, they often fail systematically, leading to constant offsets in the pose alignments (see Supp. mat.).

Another class of methods uses render-and-compare to iteratively update pose estimates [15, 18]. These tend to be more precise, but rely on good pose initialisation [15, 18]. Traditionally these have been very slow due to the time consuming rendering process and the large number of renderings required. The large number of renderings is needed as existing render-and-compare works for shape and pose estimation learn a comparison function which they directly minimise at test time using gradient descent which requires a large number of

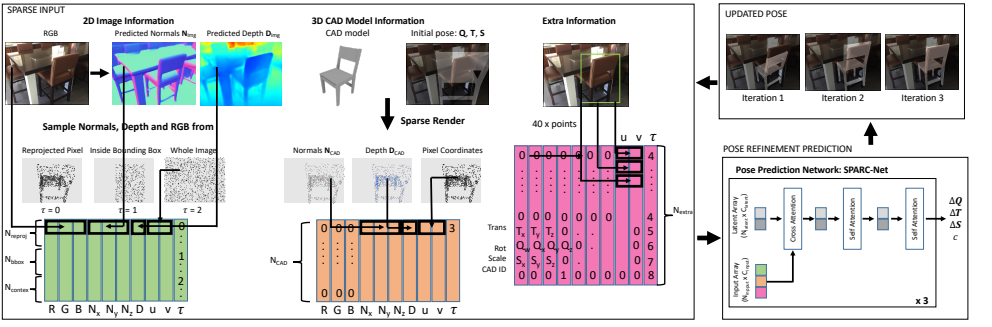

Figure 2: **Method.** (i) Given an RGB image we predict per pixel surface normals $\mathbf{N}_{Img}$ and depth $\mathbf{D}_{Img}$ (2D image information) and use bounding box and CAD model retrievals from [16] (see Sec. 3.1). (ii) We sparsely sample RGB colors, $\mathbf{N}_{Img}$ and $\mathbf{D}_{Img}$ from three different regions (the reprojected CAD model points, the bounding box and the whole image), stack them and add pixel coordinates (u,v) and a token τ allowing the network to distinguish between different input types. 3D points and surface normals are sampled from the CAD model in an initial canonical pose and reprojected into the image plane (see Sec. 3.2). 2D image information, 3D CAD model information and extra information is combined to form the input to the pose prediction network, SPARC-Net, predicting pose update steps $\Delta\mathbf{T}, \Delta\mathbf{R}, \Delta\mathbf{S}$ and rotation classification score c. Based on the prediction the pose is updated, the CAD model sparsely rendered and the next pose update is predicted (see Sec. 3.3).

steps [15, 18] (on the order of 100s to 1000). In spirit our approach is most similar to the render-and-compare approaches. However, the advantage of our method is that it does not require rendering of the full object, but just a simple reprojection of a small amount of 3D points (e.g. 100,see 5.2), and it directly learns pose updates rather than a comparison function, making our approach a lot faster than [15, 18].

Efficient transformer architectures. In recent years Transformers [34] have been used for a range of different computer vision tasks [11, 30, 32, 36]. However, the all-to-all attention mechanism used in classical transforms suffers from a quadratic scaling problem. [5, 19, 20] are one line of work aiming to make transformers more efficient and enabling them to deal with larger inputs. They achieve this by using cross-attention from the inputs to a small set of latent units and subsequently only perform all-to-all attention in this smaller latent space.

3 Method

This section explains the key steps of our method. In a first step, we perform object detection as well as surface normal and depth estimation in an image. In a second step, we sample depth and normal values from an image and combine them with reprojected depth and surface normals sampled uniformly from a CAD model. Finally, in a third step, we iterate the refinements of the 9-DoF initial pose of the CAD model, partially recomputing sparse inputs of step two for every refinement.

3.1 Object Detection, Normal and Depth Prediction

As a first step we perform 2D Object detection, including bounding box and category prediction. The category prediction determines the class of CAD model that is retrieved while the

bounding box allows us to initialise the CAD model pose (see Sec. 4) and guide the point sampling (see Sec. 3.2). We use the same object detections and CAD model retrievals as ROCA [16] for exact comparability. However, any other method could be used as well. Further we estimate per pixel surface normal $\mathbf{N}_{\text{Img}}$ and depth values $\mathbf{D}_{\text{Img}}$ as these contain crucial geometric information that can be used for precise pose predictions. For both surface normal and depth estimation, we use a light-weight convolutional encoder-decoder architecture from [1]. The training losses are the same as the state-of-the-art works [3] for surface normal estimation and [4] for depth estimation. We use ground truth surface normals provided by [17] and ground truth depth as provided by ScanNet [9] (for details see Supp. mat.). When training the surface normal and depth network the train and test split used for evaluating SPARC-Net is respected.

3.2 Input Fusion

When attempting to estimate CAD model pose one is presented with two different kinds of information: 2D image information and 3D CAD model information. We fuse the two by sampling 3D points and corresponding surface normals from the CAD model and reprojecting those into the image plane. Comparing the reprojected surface normals and the depth associated with the reprojected point to depth and surface normals estimated from the 2D image allows a network to easily evaluate the current pose and predict a pose update based on this comparison.

2D image information. We stack RGB colors and predicted surface normals $\mathbf{N}_{\text{Img}}$ and depth values $\mathbf{D}_{\text{Img}}$. However, rather than using the entire image as input we sparsely sample pixels from three selected regions. We sample N_{reproj} pixels onto which CAD model points are reprojected (see paragraph "3D CAD model information"), N_{bbox} pixels inside the bounding box and N_{context} pixels from the entire image. Respectively, these provide information on whether the current pose matches the image, what the pose update should be and global context. In Fig. 4b and Tab. 2 we demonstrate that sampling image information sparsely reduces overfitting, leading to a better alignment accuracy, and greatly reducing memory consumption and network complexity (see Sec. 5.2). For each of the sampled pixels containing color, depth and surface normal values we add the pixel coordinates (u,v) and a token τ (to allow the network to distinguish between different types of inputs) and stack them to a $(N_{\text{reproj}}+N_{\text{bbox}}+N_{\text{context}})\times(3+3+1+2+1)$ dimensional tensor as seen in Fig. 2.

3D CAD model information. We represent the CAD model as a collection of N_{CAD} 3D points sampled uniformly from the object surface. We simply reproject 3D object points and their surface normals into the image plane which reduces the rendering operation to a perspective projection using a single matrix multiplication. The intuition behind this idea is that for texture-less CAD models almost all information is contained in the 3D shape. Hence there is no need for a full rendering pipeline that takes into account material, lighting, textures or visibility[1]. We stack reprojected surface normals $\mathbf{N}_{\text{CAD}}$ and the computed depth values $\mathbf{D}_{\text{CAD}}$ with their pixel coordinates and a token $\tau=3$. Padding this tensor with zeros as RGB values we obtain a $N_{\text{CAD}}\times 10$ shaped tensor.

Extra information. Additionally, we encode the bounding box information by sampling 10 points from each of the sides of the bounding box and inputting their pixel values with a different token $\tau=4$ (padding extra dimensions with 0's). We also directly input the ini-

[1] While reprojecting all CAD model points into the image plane ignores potentially important occlusion effects we found SPARC-Net to perform well regardless. We have also tried to add information about whether a 3D object point is occluded or not, but did not observe any improvement in performance.

tialised pose consisting of translation $\mathbf{T}$, rotation parameterised as quaternion $\mathbf{Q}$, scale $\mathbf{S}$ and the CAD model ID encoded as a binary vector, each padded with 0's and a unique token. Finally, we concatenate the three different blocks containing image, reprojected CAD model and extra information to form a $(N_{\text{reproj}} + N_{\text{bbox}} + N_{\text{context}} + N_{\text{CAD}} + N_{\text{extra}}) \times 10 = (N_{\text{input}} \times 10)$ dimensional tensor. This tensor is Fourier encoded with 64 frequency bands with a maximum frequency of 1120 (same as original Perceiver [19] applied to point clouds) resulting in a $N_{\text{input}} \times (10 + 64 * 2 + 1) = N_{\text{input}} \times C_{\text{input}}$ tensor which serves as the network input described in the next section.

3.3 Pose updates and Iterative Refinement

Combining image information and CAD model information in the image plane allows the network to easily evaluate a given pose and predict a refinement based on this evaluation. We repeat this process N_{iter} times allowing the network to refine its own predictions. Specifically, given an initial pose $(\mathbf{T}, \mathbf{Q}, \mathbf{S})$ and all combined information explained in Sec. 3.2 we use a Perceiver network [19] to predict refinement steps $(\Delta\mathbf{T}, \Delta\mathbf{Q}, \Delta\mathbf{S})$ such that $(\mathbf{T} + \Delta\mathbf{T}, \mathbf{Q} \cdot \Delta\mathbf{Q}, \mathbf{S} + \Delta\mathbf{S})$ is close to the correct pose $(\mathbf{T}_{\text{gt}}, \mathbf{Q}_{\text{gt}}, \mathbf{S}_{\text{gt}}) = (\mathbf{T} + \Delta\mathbf{T}_{\text{gt}}, \mathbf{Q} \cdot \Delta\mathbf{Q}_{\text{gt}}, \mathbf{S} + \Delta\mathbf{S}_{\text{gt}})$. Similar to previous works [22, 23] we find that learning rotations over the full, non-euclidean rotation space is difficult. We therefore follow a coarse-to-fine approach where we simultaneously train the network to predict a binary classification c whether an initial rotation lies within the correct 90° rotation bin c_{gt} around the vertical. For classifying rotation we use a standard binary cross-entropy loss L_{BCE} and for learning the offsets we use the L2 loss such that our loss function is given by:

$$L_{\text{align}} = w_{\text{c}} L_{\text{BCE}}(c_{\text{gt}}, c) + w_{\text{t}} L_{\text{L2}}(\Delta\mathbf{T}_{\text{gt}}, \Delta\mathbf{T}) + w_{\text{s}} L_{\text{L2}}(\Delta\mathbf{S}_{\text{gt}}, \Delta\mathbf{S}) + w_{\text{q}} L_{\text{L2}}(\Delta\mathbf{Q}_{\text{gt}}, \Delta\mathbf{Q}). \quad (1)$$

At test time we apply SPARC-Net to four rotation initialisation that are 90° rotated around the vertical. For each of those we predict the probability c indicating whether the correct rotation lies within ±45° of the tested initialisation. For the initialisation with the highest estimated probability c we subsequently predict the 9 DoF pose updates and iteratively rerender and refine the pose.

4 Experimental Setup

This section briefly describes the dataset used for training and testing our method, as well as the evaluation metrics used and the hyperparameters chosen.

ScanNet dataset. Similar to [16, 22, 23, 27] we train and test our approach on the ScanNet25k image data [9] for which [2] provide CAD model annotation for a wide range of objects. This dataset contains 20k training images representing 1200 train scenes and 5k validation images from 300 different evaluation scenes. We train and test our method on the 9 categories with the most CAD model annotations covering more than 2500 distinct shapes.

Evaluation metric. We follow the original evaluation protocol introduced by Scan2CAD [2] which evaluates CAD model alignments per-scene. In the same way as ROCA [16] we transform predicted CAD model poses into ScanNet [9] world coordinates and apply 3D non-maximum suppression to eliminate multiple detections of the same object from different images. A CAD model prediction is considered 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 ratio is less than 20%. We found that there was a bug in the original evaluation code introduced by [2] which was subsequently used for evaluating [16, 22]. When computing the scale error

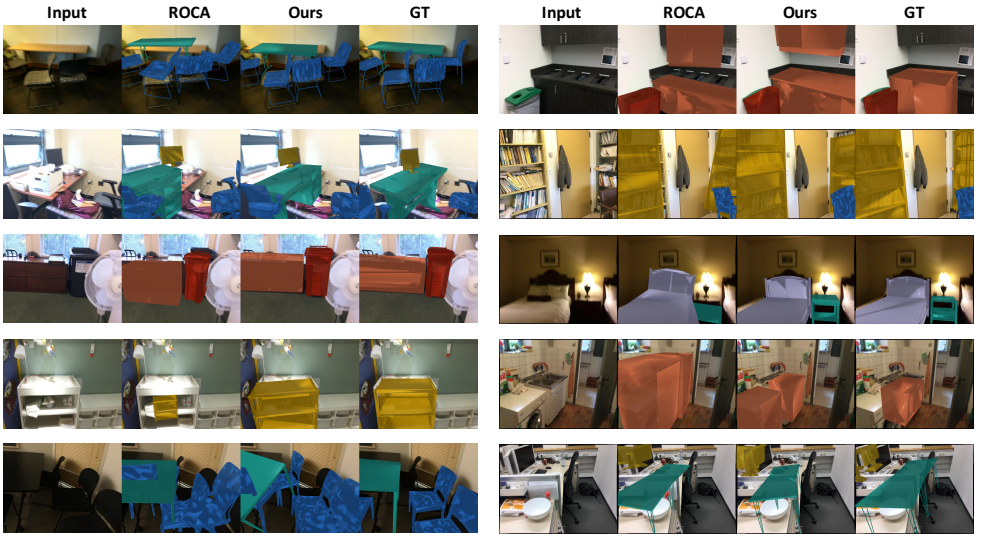

Figure 3: **Qualitative results on ScanNet [9].** Our sparse, render-and-compare approach allows for more precise CAD model alignment compared to ROCA [16]. Typical failure cases of ROCA include systematic failure when predicting normalised object coordinates leading to wrong object translations (row 3) and wrong scale predictions (row 4). Row 5 shows failure cases for both methods in complex scenes (left) and when object boundaries are not clearly identifiable (right).

the formula $s_{\text{error}} = |\sum_{i=x,y,z}(S_i/S_i^{\text{gt}} - 1)|$ was used instead of $s_{\text{error}} = \sum_{i=x,y,z}|(S_i/S_i^{\text{gt}}) - 1|$ which allowed scale errors in different directions to cancel each other out. We corrected this mistake and reevaluated [16, 22] (see Supp. mat.).

Input details. For the main experiment we choose $N_{\text{CAD}} = 1000$ and use the reprojected points as samples from the 2D image such that $N_{\text{reproj}} = 1000$. Further we use $N_{\text{bbox}} = 1000$ and $N_{\text{context}} = 5000$ as also shown in Fig. 2.

Perceiver architecture. For the Perceiver [19] we set $N_{\text{latent}} = 128$ and $C_{\text{latent}} = 256$ and as depicted in Fig. 2 repeat three blocks of one cross attention layer followed by two self-attention layers with weight sharing between each of the layers in the three blocks.

Train details. We train the SPARC-Net for 300 epochs using the LAMB [37] optimiser (as used by the original Perceiver [19]) with learning rate 0.001. For the loss function in Eq. 1 we set $w_c = 0.5$, $w_t = 0.5$, $w_s = 0.5$ and $w_q = 1$. We use a batchsize of 80. For more information on sampling and initialising poses at train and test time see the Supp. mat.

Implementation Details. All code is implemented in PyTorch [31]. SPARC-Net was trained on a single TitanXp for 36 hours.

5 Experimental Results

In the first part of this section we compare ourselves to the state-of-the-art approaches [16, 22, 27]. In the second part, we investigate and ablate different key aspects of our system, including the 2D pixel sampling, the sparse 3D object representation, render-and-compare, the number of refinements used and the relevance of different input information.

Method	bathtub	bed	bin	bkshlf	cabinet	chair	display	sofa	table	**class**	**instance**
Number of Instances #	120	70	232	212	260	1093	191	113	553	9	2844
Total3D-ODN [7]	10.0	2.9	16.8	2.8	4.2	14.4	13.1	5.3	6.7	8.5	10.4
Mask2CAD-b5 [2]	7.5	2.9	24.6	1.4	5.0	29.9	13.1	5.3	5.6	10.6	16.7
ROCA [16]	20.8	8.6	26.3	9.0	13.1	39.9	**24.6**	10.6	12.7	18.4	25.0
SPARC-Net (ours)	**25.8**	25.7	24.6	**14.2**	**20.8**	51.5	17.8	**28.3**	15.4	24.9	31.8
SPARC-Net + ROCA rot init	25.0	**30.0**	**36.2**	**14.2**	19.2	**52.3**	20.4	**28.3**	**20.1**	**27.3**	**34.1**

Table 1: **Alignment Accuracy on ScanNet** [2, 9] in comparison to the state-of-the-art.

5.1 Main Results

Tab. 1 shows that our approach outperforms all competing approaches in all object classes (except for ROCA on the class "display"). Further we significantly improve both the class average alignment accuracy from 18.4% to 24.9% as well as the instance alignment accuracy from 25.0% to 31.8%. Visually comparing our predictions to ROCA [16] in Fig. 3 we note that the improvements are mainly due to better translation and scale predictions. ROCA [16] relies on predicting normalised object coordinates for estimating translation. However, due to difficulties in aligning and registering different CAD models to a canonical frame, predicting NOCs often fails with a systematic offset leading to an offset in the final alignment (see Supp. mat.). In contrast, our approach is independent of the existence of a canonical object frame as it directly compares a reprojected object to an image. Further, ROCA decouples the scale predictions from predicting rotation and translation which can produce very wrong alignments for bad scale predictions (see Fig. 3). We also evaluate our system when using ROCA's rotation predictions as an initialisation as opposed to our own classified rotation bins. Here we observe a further improvement of our alignment accuracy. The reason why ROCA's rotation predictions are good is that they are computed geometrically from correspondences and are largely unaffected by systematic offsets in NOCs predictions and wrong scale predictions.

5.2 Ablation

Note that in addition to average class alignment accuracy and instance alignment accuracy we report $\mathbf{T},\mathbf{R},\mathbf{S}$ and c accuracy in Tab. 2 where the same evaluation protocol is used as described in Sec. 4, but rather than requiring $\mathbf{T}$, $\mathbf{R}$ and $\mathbf{S}$ to be correct at the same time only the quantity of interest is required to be correct (c is considered correct if the binary rotation classification is correct). If not otherwise specified in the following "accuracy" or "performance" refers to instance alignment accuracy.

Image sampling. We compare sampling the entire image to sparse sampling and observe worse alignment accuracy when using the entire image (25.0% vs. 31.8%) due to faster overfitting at train time (see Fig. 4b). We further simplify our sampling scheme and instead of sampling the image globally, the bounding box and the reprojected query, we use samples from just the bounding box, $N_{\text{bbox}} = 1000$. Surprisingly, we observe that we can recover almost the same alignment accuracy 30.1% compared to 31.8% before. By traininig and testing with less than 1% of the original number of pixel (image resolution 480 x 360) we reduce the memory by a factor of 100. The complexity of the Perceiver [19] is given by $\mathcal{O}(N_{\text{input}}N_{\text{latent}}) + \mathcal{O}(LN^2_{\text{latent}}) \approx \mathcal{O}(N_{\text{input}}N_{\text{latent}})$ where $N_{\text{input}} >> N_{\text{latent}}$, $N_{\text{input}} >> L$. When using the whole image as input N_{input} is dominated by the number of pixels $N_{\text{input}} \approx 480 \times 360$. Therefore as $N_{\text{latent}} = 128$ and $L = 2$ (two self-attention layers following the cross-attention layer) sparse inputs also reduce the compute complexity by a factor of 100, greatly speeding up training and testing.

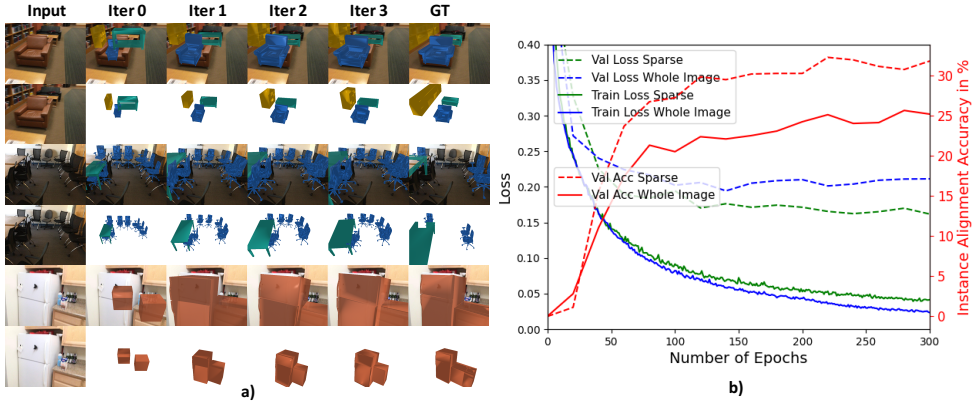

Figure 4: a) **Visualisation of refinements** in iterative render-and-compare. b) **Train and validation loss as well as validation accuracy** on the ScanNet dataset [7, 9]. When training on the whole image SPARC-Net overfits the training data more easily resulting in higher validation loss and worse instance alignment accuracy.

Sparse CAD model representation. When reducing the number of points N_{CAD} we observe that remarkably with just 100 points we obtain similar performance as with 1000 points (30.0% vs. 31.8%), showing that 3D shapes can indeed be represented very sparsely. However, as we further decrease the number of points we observe a rapid decline in performance (20.9% for $N_{\text{CAD}} = 50$ and 4.2% for $N_{\text{CAD}} = 50$. Interestingly, this decrease is almost entirely due to worse rotation predictions (see **R** accuracy 24.8% compared to 67.9%). Intuitively, this makes sense as an extremely sparse point cloud has very little structure from which to predict rotation while translation and scale can still be predicted. Initialising those extremely sparse models with ROCA [16] rotation predictions we can still recover reasonable alignment accuracy of 24.0% and 29.9% for $N_{\text{CAD}} = 20$ and $N_{\text{CAD}} = 50$.

Render-and-Compare. We test the assumption that Render-and-Compare is beneficial for precise pose predictions by considering two networks which are trained either on no 3D shape information but just the CAD model ID or for which the CAD model is always presented in the same canonical pose. The first is motivated by works (e.g. [22, 23]) that contain no explicit 3D shape information and the second by [21] that perform iterative pose updates without explicit rerendering. Similar to before both of these networks fail to learn accurate rotation predictions (both for the classification and the refinement) therefore we only show results where we initialise poses with ROCA [16] rotation predictions. Here we note that using just the CAD ID results in a significant decrease in performance (21.0%), whereas using 3D information but presenting the CAD model in the same canonical pose achieves 25.9% instance alignment accuracy.

Number of refinements. Training and evaluating our network for different number of iterations we observe a significant improvement when increasing the number of refinements from 1 (26.4%) to 3 (31.8%) and no more improvement for 5 refinements (31.3%).

Input modalities. Furthermore, we investigate the importance of the different information that is provided as input to the network. Here we find that both depth and normal predictions are crucial for SPARC-Net. Not providing depth predictions from the 2D image reduces the accuracy to 18.8% while not providing normal predictions reduces it to 20.7%. We find that

Accuracy	2D Image Sampling					3D CAD model points						R. and C.		N Refinement			Input Information			
	$N_{\text{reproj}} = 100$ $N_{\text{bbox}} = 100$ $N_{\text{context}} = 0$ $N_{\text{CAD}} = 100$	$N_{\text{reproj}} = 0$ $N_{\text{bbox}} = 200$ $N_{\text{context}} = 0$ $N_{\text{CAD}} = 100$	$N_{\text{reproj}} = 0$ $N_{\text{bbox}} = 1000$ $N_{\text{context}} = 0$ $N_{\text{CAD}} = 100$	$N_{\text{reproj}} = 0$ $N_{\text{bbox}} = 0$ $N_{\text{context}} = 5000$ $N_{\text{CAD}} = 1000$	Whole img $N_{\text{CAD}} = 1000$	20	20*	50	50*	100	1000	Just CAD ID*	Canonical pose*	1	3	5	No D	No N	No RGB	No RTS or CAD info
Class	12.9	12.7	23.9	14.8	20.3	3.5	17.7	15.9	24.9	23.1	24.9	15.0	20.2	16.6	24.9	25.2	15.0	16.6	24.7	22.2
Instances	18.5	19.7	**30.1**	20.1	**25.0**	**4.2**	**24.0**	**20.9**	**29.9**	**30.0**	**31.8**	**21.0**	**25.9**	**26.4**	**31.8**	**31.3**	**18.8**	**20.7**	**30.5**	**28.9**
T	40.8	42.1	42.4	37.1	38.7	38.0	39.9	38.2	42.5	42.2	**45.4**	36.4	41.4	42.3	**45.4**	43.7	**30.7**	40.2	42.6	**42.2**
R	49.1	48.7	70.8	61.4	68.2	**24.8**	63.7	58.4	70.9	71.1	**67.9**	63.8	63.7	65.5	**67.9**	71.5	68.2	**56.8**	72.3	66.3
S	61.0	62.4	69.0	61.3	66.8	63.7	63.3	66.4	67.7	67.8	68.4	62.9	68.9	62.8	68.4	66.4	64.5	63.7	68.5	67.2
R class	70.7	71.3	75.2	69.7	72.9	53.4	76.4	68.9	77.4	75.6	75.9	77.0	76.3	74.8	75.9	75.6	73.0	74.5	76.1	74.4

Table 2: **Ablation.** We perform a number of different ablations as explained in detail in Sec. 5.2. Bold numbers are the ones referred to in Sec. 5.2. Note that a "*" indicates that ROCA [16] rotation predictions are used for initialising rotations. Our main experiment is listed twice under "3D CAD model points $N_{\text{CAD}} = 1000$" and "N Refinements, $N = 3$" for convenience. "R. and C." stands for Render-and-Compare.

for depth this drop in performance is mainly due to worse **T** predictions (30.7% compared to 45.4%) whereas for the normals it is due to worse rotation predictions (56.8% compared to 67.9%) which is exactly as one would expect. In contrast, not providing RGB color only reduces the accuracy marginally to 30.5%. This makes sense as given depth and surface normal estimates color only adds very little information for shape alignments. When no extra information is provided the accuracy is reduced to 28.9%. The individual accuracies suggest that the network uses extra information to learn the distribution of object positions as we observe the largest decrease for **T** accuracy to 42.2%.

6 Conclusion

We have proposed SPARC, a sparse render-and-compare approach for CAD model alignment from a single RGB image. We show how to effectively combine 3D CAD model information and 2D image information with early-fusion which helps our pose prediction network to estimate accurate pose updates. In this way we improve the state-of-the-art on ScanNet [9] from 25.0% to 31.8% instance alignment accuracy. We demonstrate that using sparse 2D image information (less than 1% of available pixels) reduces overfitting leading to better pose predictions and greatly reducing memory and network complexity. Furthermore, sparse input processing may bring even more benefits when dealing with sensory input that is naturally sparse such as LIDAR or event cameras. In the future, we plan to expand this work to updating not only the estimated pose, but also the CAD model shape in order to alleviate the requirement of CAD models that are very similar to the objects in the image.

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] 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.

[4] 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.

[5] João Carreira, Skanda Koppula, Daniel Zoran, Adrià Recasens, Catalin Ionescu, Olivier J. Hénaff, Evan Shelhamer, Relja Arandjelovic, Matthew M. Botvinick, Oriol Vinyals, Karen Simonyan, Andrew Zisserman, and Andrew Jaegle. Hierarchical perceiver. *CoRR*, abs/2202.10890, 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] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction. In *Proc. 14th European Conference on Computer Vision*, Amsterdam, NL, October 2016.

[9] 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.

[10] 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.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[12] Francis Engelmann, Konstantinos Rematas, Bastian Leibe, and Vittorio Ferrari. From Points to Multi-Object 3D Reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[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] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. *ICLR*, 2022.

[21] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[22] 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.

[23] 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.

[24] 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.

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

[26] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proc. 16th European Conference on Computer Vision*, Glasgow, UK (Virtual), August 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] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep Mesh Reconstruction from Single RGB Images via Topology Modification Networks. In *Proc. IEEE Int. Conf. on Computer Vision*, Seoul, Korea, October 2019.

[29] 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.

[30] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064. PMLR, 10–15 Jul 2018.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[32] Chen Sun, Austin Myers, Carl Vondrick, Kevin P. Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7463–7472, 2019.

[33] Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 103–110, 2012. doi: 10.1109/CVPR.2012.6247664.

[34] 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.

[35] 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.

[36] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)*, pages 106–115. IEEE, 2021.

[37] 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.

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